

George I. Davida
Yair Frankel (Eds.)

LNCS 2200

Information Security

4th International Conference, ISC 2001
Malaga, Spain, October 2001
Proceedings



Springer

Lecture Notes in Computer Science

2200

Edited by G. Goos, J. Hartmanis, and J. van Leeuwen

Springer

Berlin

Heidelberg

New York

Barcelona

Hong Kong

London

Milan

Paris

Tokyo

George I. Davida Yair Frankel (Eds.)

Information Security

4th International Conference, ISC 2001
Malaga, Spain, October 1-3, 2001
Proceedings



Springer

Series Editors

Gerhard Goos, Karlsruhe University, Germany
Juris Hartmanis, Cornell University, NY, USA
Jan van Leeuwen, Utrecht University, The Netherlands

Volume Editors

George I. Davida
University of Wisconsin-Milwaukee, Department of EECS
Milwaukee, WI 53201, USA
E-mail: davida@cs.uwm.edu

Yair Frankel
Techtegrity, LLC
122 Harrison, Westfield NJ 07090, USA
E-mail: yfrankel@cryptographers.com

Cataloging-in-Publication Data applied for

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Information security : 4th international conference ; proceedings / ISC
2001, Malaga, Spain, October 1 - 3, 2001. George I. Davida ; Yair Frankel
(ed.). - Berlin ; Heidelberg ; New York ; Barcelona ; Hong Kong ; London ;
Milan ; Paris ; Tokyo : Springer, 2001
(Lecture notes in computer science ; Vol. 2200)
ISBN 3-540-42662-0

CR Subject Classification (1998): E.3, D.4.6, K.6.5, F.2.1, C.2, J.1, C.3

ISSN 0302-9743

ISBN 3-540-42662-0 Springer-Verlag Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer-Verlag. Violations are liable for prosecution under the German Copyright Law.

Springer-Verlag Berlin Heidelberg New York
a member of BertelsmannSpringer Science+Business Media GmbH

<http://www.springer.de>

© Springer-Verlag Berlin Heidelberg 2001
Printed in Germany

Typesetting: Camera-ready by author, data conversion by PTP-Berlin, Stefan Sossna
Printed on acid-free paper SPIN: 10840698 06/3142 5 4 3 2 1 0

Preface

The Information Security Conference 2001 brought together individuals involved in multiple disciplines of information security to foster the exchange of ideas. The conference, an outgrowth of the Information Security Workshop (ISW) series, was held in Málaga, Spain, on October 1–3, 2001. Previous workshops were ISW '97 at Ishikawa, Japan; ISW '99 at Kuala Lumpur, Malaysia; and ISW 2000 at Wollongong, Australia. The General Co-chairs, Javier López and Eiji Okamoto, oversaw the local organization, registration, and performed many other tasks.

Many individuals deserve thanks for their contribution to the success of the conference. José M. Troya was the Conference Chair. The General Co-chairs were assisted with local arrangements by Antonio Maña, Carlos Maraval, Juan J. Ortega, José M. Sierra, and Miguel Soriano.

This was the first year that the conference accepted electronic submissions. Many thanks to Dawn Gibson for assisting in developing and maintaining the electronic submission servers. The conference received 98 submissions of which 37 papers were accepted for presentation. These proceedings contain revised versions of the accepted papers. Revisions were not checked and the authors bear full responsibility for the contents of their papers.

The Program Committee consisted of Elisa Bertino, Università di Milano; G. R. Blakely, Texas A&M University; John Daugman, Cambridge University; Jorge Dávila, Polytechnic Univ. of Madrid; Giovanni DiCrescenzo, Telcordia; Josep Domingo-Ferrer, Univ. Rovira i Virgili; Dieter Gollmann, Microsoft Research; Sigrid Guergens, GMD; Hiroaki Kikuchi, Tokai University; Chi-Sung Lai, Natl. Cheng Kung Univ.; Wenbo Mao, HP Laboratories; Masahiro Mambo, Tohoku University; Catherine Meadows, NRL; Sang-Jae Moon, Kyungpook Natl. University; Yuko Murayama, Iwate Prefectural University; René Peralta, Yale University; Josef Pieprzyk, University of Wollongong; Sihan Qing, Chinese Academy of Sciences; Susie Thomson, Datacard; Routo Terada, Univ. of S. Paulo; Yiannis Tsiounis, InternetCash; Moti Yung, Certco; Yuliang Zheng, Monash University; Jianying Zhou, Oracle Corp. Members of the committee spent numerous hours in reviewing papers and providing advice. The committee was also assisted by our colleagues: Marc Alba, Clemente Galdi, Sang-Wook Kim, Yi Mu, Anna Oganian, Francesc Sebé, Rajan Shankaran, Igor Shparlinski. We apologize for any inadvertent omissions. Our thanks to the program committee and all reviewers.

We thank all the authors who submitted papers to this conference. Without their submissions this conference could not have been a success.

July 2001

George I. Davida
Yair Frankel

Information Security Conference 2001

October 1–3, 2001, Málaga, Spain

Conference Chair

José M. Troya, University of Málaga (Spain)

General Co-chairs

Javier López, University of Málaga (Spain)

Eiji Okamoto, Toho University (Japan)

Program Co-chair

George I. Davida, University of Wisconsin-Milwaukee (USA)

Yair Frankel, TechTegrity L.L.C. (USA)

Program Committee

Elisa Bertino Università di Milano (Italy)
G. R. Blakely..... Texas A&M University (USA)
John DaugmanCambridge University (UK)
Jorge Dávila Polytechnic Univ. of Madrid (Spain)
Giovanni DiCrescenzo Telcordia (USA)
Josep Domingo-Ferrer..... Univ. Rovira i Virgili (Spain)
Dieter Gollmann.....Microsoft Research (UK)
Sigrid GuergensGMD (Germany)
Hiroaki KikuchiTokai University (Japan)
Chi-Sung Laih.....Natl. Cheng Kung Univ. (Taiwan)
Wenbo Mao.....HP Laboratories (UK)
Masahiro Mambo Tohoku University (Japan)
Catherine Meadows.....NRL (USA)
Sang-Jae MoonKyungpook Natl. University (Korea)
Yuko MurayamaIwate Prefectural University (Japan)
Rene Peralta Yale University (USA)
Josef Pieprzyk University of Wollongong (Australia)
Si-han Qing Chinese Academy of Sciences (China)
Susie Thomson..... Datacard (UK)
Routo Terada..... Univ. of S. Paulo (Brazil)
Yiannis Tsiounis InternetCash (USA)
Moti Yung..... CertCo (USA)
Yuliang ZhengMonash University (Australia)
Jianying Zhou Oracle Corp. (USA)

Table of Contents

Key Distribution

Bounds and Constructions for Unconditionally Secure Distributed Key Distribution Schemes for General Access Structures	1
<i>Carlo Blundo, Paolo D'Arco (Università di Salerno)</i> <i>Vanessa Daza, and Carles Padró (Universitat Politècnica de Catalunya)</i>	
Privacy Amplification Theorem for Noisy Main Channel	18
<i>Valeri Korjik, Guillermo Morales-Luna (CINVESTAV-IPN),</i> <i>and Vladimir B. Balakirsky (EIDMA)</i>	

Protocols

Efficient Kerberized Multicast in a Practical Distributed Setting.....	27
<i>Giovanni Di Crescenzo (Telcordia Technologies)</i> <i>and Olga Kornievskaja (University of Michigan)</i>	
Suitability of a Classical Analysis Method for E-commerce Protocols.....	46
<i>Sigrid Gürgens (GMD-SIT) and Javier Lopez (University of Malaga)</i>	

Enhancing Technologies

Hypocrates (A New Proactive Password Checker)	63
<i>Carlo Blundo, Paolo D'Arco, Alfredo De Santis,</i> <i>and Clemente Galdi (Università di Salerno)</i>	
Lenient/Strict Batch Verification in Several Groups	81
<i>Fumitaka Hoshino, Masayuki Abe,</i> <i>and Tetsutaro Kobayashi (NTT Corporation)</i>	

Privacy

Absolute Privacy in Voting	95
<i>Dmitri Asonov (Humboldt-Universität zu Berlin),</i> <i>Markus Schaal (Technische Universität Berlin), and</i> <i>Johann-Christoph Freytag (Humboldt-Universität zu Berlin)</i>	
A Logical Model for Privacy Protection	110
<i>Tsan-sheng Hsu, Churn-Jung Liao,</i> <i>and Da-Wei Wang (Academia Sinica)</i>	

Software Protection

DISSECT: DIStribution for SECurity Tool	125
<i>Enriquillo Valdez (Polytechnic University of New York)</i> <i>and Moti Yung (CertCo, Inc.)</i>	
An Approach to the Obfuscation of Control-Flow of Sequential Computer Programs	144
<i>Stanley Chow, Yuan Gu, Harold Johnson (Cloakware Corporation),</i> <i>and Vladimir A. Zakharov (Moscow State University)</i>	

Message Hiding I

A Practical and Effective Approach to Large-Scale Automated Linguistic Steganography	156
<i>Mark Chapman (Omni Tech Corp.),</i> <i>George I. Davida (University of Wisconsin-Milwaukee), and</i> <i>Marc Rennhard (Swiss Federal Institute of Technology)</i>	
Robust New Method in Frequency Domain Watermarking	166
<i>David Sánchez, Agustín Orfila, Julio César Hernández,</i> <i>and José María Sierra (Carlos III University)</i>	

PKI Issues and Protocols

On the Complexity of Public-Key Certificate Validation	183
<i>Diana Berbecaru, Antonio Lioy, and</i> <i>Marius Marian (Politecnico di Torino)</i>	
Liability of Certification Authorities: A Juridical Point of View	204
<i>Apol·lònia Martínez-Nadal and</i> <i>Josep L. Ferrer-Gomila (Universitat de les Illes Balears)</i>	

Hardware Implementations

Experimental Testing of the Gigabit IPSec-Compliant Implementations of Rijndael and Triple DES Using SLAAC-1V FPGA Accelerator Board	220
<i>Pawel Chodowiec, Kris Gaj (George Mason University),</i> <i>Peter Bellows, and Brian Schott (University of Southern California)</i>	
Elliptic Curve Arithmetic Using SIMD	235
<i>Kazumaro Aoki (NTT Communications), Fumitaka Hoshino,</i> <i>Tetsutaro Kobayashi, and Hiroaki Oguro (NTT Corporation)</i>	

On the Hardware Implementation of the 3GPP Confidentiality and Integrity Algorithms	248
<i>Kostas Marinis (National Technical University of Athens), Nikos K. Moshopoulos, Fotis Karoubalis (Atmel Hellas), and Kiamal Z. Pekmestzi (National Technical University of Athens)</i>	
Efficient Implementation of Elliptic Curve Cryptosystems on an ARM7 with Hardware Accelerator	266
<i>Sheng-Bo Xu and Lejla Batina (Securealink B.V.)</i>	

Cryptanalysis and Prevention

A Theoretical DPA-Based Cryptanalysis of the NESSIE Candidates FLASH and SFLASH.....	280
<i>Rainer Steinwandt, Willi Geiselmann, and Thomas Beth (Universität Karlsruhe)</i>	
Quadratic Relations for S-Boxes: Their Minimum Representations and Bounds	294
<i>Routo Terada and Paulo G. Pinheiro (University of S. Paulo)</i>	
Approximate Power Roots in \mathbb{Z}_m	310
<i>Ismael Jiménez-Calvo (C.S.I.C.) and German Sáez-Moreno (Universitat Politècnica de Catalunya)</i>	
Securing Elliptic Curve Point Multiplication against Side-Channel Attacks	324
<i>Bodo Möller (Technische Universität Darmstadt)</i>	

Implementations

A Flexible Role-Based Access Control Model for Multimedia Medical Image Database Systems	335
<i>Sofia Tzelepi and George Pangalos (Aristotelian University)</i>	
A Secure Publishing Service for Digital Libraries of XML Documents	347
<i>Elisa Bertino, Barbara Carminati (Università di Milano), and Elena Ferrari (Università dell'Insubria)</i>	

Non-repudiation Techniques

An Optimistic Non-repudiation Protocol with Transparent Trusted Third Party	363
<i>Olivier Markowitch and Steve Kremer (Université Libre de Bruxelles)</i>	

Persistent Authenticated Dictionaries and Their Applications	379
<i>Aris Anagnostopoulos (Brown University), Michael T. Goodrich (University of California), and Roberto Tamassia (Brown University)</i>	

Contracts and Auctions

Efficient Optimistic N-Party Contract Signing Protocol	394
<i>Josep L. Ferrer-Gomila, Magdalena Payeras-Capellà, and Llorenç Huguet-Rotger (Universitat de les Illes Balears)</i>	
Efficient Sealed-Bid Auctions for Massive Numbers of Bidders with Lump Comparison	408
<i>Koji Chida, Kunio Kobayashi, and Hikaru Morita (NTT Corporation)</i>	

Message Hiding II

Oblivious Image Watermarking Robust against Scaling and Geometric Distortions	420
<i>Francesc Sebé and Josep Domingo-Ferrer (Universitat Rovira i Virgili)</i>	
Fingerprinting Text in Logical Markup Languages	433
<i>Christian D. Jensen (Trinity College Dublin)</i>	

Payments

SPEED Protocol: Smartcard-Based Payment with Encrypted Electronic Delivery	446
<i>Antonio Ruiz, Gregorio Martínez, Oscar Cánovas, and Antonio F. Gómez (University of Murcia)</i>	
Efficient Transferable Cash with Group Signatures	462
<i>Ik Rae Jeong, Dong Hoon Lee, and Jong In Lim (Korea University)</i>	

Security Applications

An Auditable Metering Scheme for Web Advertisement Applications	475
<i>Liqun Chen and Wenbo Mao (Hewlett-Packard Laboratories)</i>	

Broker-Based Secure Negotiation of Intellectual Property Rights	486
<i>Jaime Delgado (Universitat Pompeu Fabra), Isabel Gallego</i>	
<i>(Universitat Politècnica de Catalunya), and Xavier Perramon</i>	
<i>(Universitat Pompeu Fabra)</i>	

Network and OS Security

Design of the Decision Support System for Network Security	
Management to Secure Enterprise Network	497
<i>Jae Seung Lee (ETRI) and</i>	
<i>Sang Choon Kim (Samchok National University)</i>	

Measuring False-Positive by Automated Real-Time Correlated Hacking	
Behavior Analysis	512
<i>Jia Wang and Insup Lee (University of Pennsylvania)</i>	

Design of UNIX System for the Prevention of Damage Propagation	
by Intrusion and Its Implementation Based on 4.4BSD	536
<i>Kenji Masui, Masahiko Tomoishi,</i>	
<i>and Naoki Yonezaki (Tokyo Institute of Technology)</i>	

Author Index	553
---------------------------	------------

Bounds and Constructions for Unconditionally Secure Distributed Key Distribution Schemes for General Access Structures^{*}

Carlo Blundo¹, Paolo D'Arco¹, Vanesa Daza², and Carles Padró²

¹ Dipartimento di Informatica ed Applicazioni
Università di Salerno, 84081 Baronissi (SA), Italy
{carlu, paodar}@dia.unisa.it

² Departament de Matemàtica Aplicada IV
Universitat Politècnica de Catalunya, 08034 Barcelona, Spain
{vdaza, matcpl}@mat.upc.es

Abstract. In this paper we investigate the issues concerning with the use of a single server across a network, the *Key Distribution Center*, to enable private communications within groups of users. After providing several motivations, showing the advantages related to the *distribution* of the task accomplished by this server, we describe a model for such a distribution, and present bounds on the amount of resources required in a real-world implementation: random bits, memory storage, and messages to be exchanged. Moreover, we introduce a linear algebraic approach to design optimal schemes distributing a Key Distribution Center and we show that some known previous constructions belong to the proposed framework.

Keywords: Key Distribution, Protocols, Distributed Systems.

1 Introduction

Secure communications over insecure channels can be carried out using encryption algorithms. If a public key infrastructure is available, public key algorithms can be employed. However, in this setting, if a user wishes to send the same message to n different users, he has to compute n encryptions of the message using n different public keys, and he has to send the message to each of them. Moreover, public key encryption and decryption are slow operations and, when the communication involves a group of users, hereafter referred to as a *conference*, this communication strategy is completely inefficient from a computational and communication point of view as well.

An improvement on the “trivial” use of public key algorithms can be the *hybrid* approach: a user chooses at random a key and sends it, in encrypted form (public key), to all the other members of the conference. Then, they can securely

^{*} The work of the third and the fourth authors was partially supported by Spanish *Ministerio de Ciencia y Tecnología* under project TIC 2000-1044.

communicate using a symmetric algorithm. Indeed, symmetric encryption algorithms are a few orders of magnitude more efficient than public key ones. Triple-DES, RC6, and RIJNDAEL, for example, are fast algorithms, spreadly used, and supposed to be secure. Besides, if a broadcast channel is available, a message for different recipients needs to be sent just once. Hence, better performances can be achieved with symmetric algorithms.

However, the hybrid protocol described before is still not efficient, and it is possible to do better. Actually, the question is how can be set up an *efficient* protocol to give each conference a key.

A common solution is the use of a Key Distribution Center (KDC, for short), a server responsible of the distribution and management of the secret keys. The idea is the following. Each user shares a common key with the center. When he wants to securely communicate with other users, he sends a request for a conference key. The center checks for membership of the user in that conference, and distributes in encrypted form the conference key to each member of the group. Needham and Schroeder [20] began this approach, implemented most notably in the Kerberos System [21], and formally defined and studied in [1], where it is referred to as the *three party model*.

The scheme implemented by the Key Distribution Center to give each conference a key is called a *Key Distribution Scheme* (KDS, for short). The scheme is said to be *unconditionally secure* if its security is independent from the computational resources of the adversaries.

Several kinds of Key Distribution Schemes have been considered so far: Key Pre-Distribution Schemes (KPSs, for short), Key Agreement Schemes (KASs, for short) and Broadcast Encryption Schemes (BESs, for short) among others. The notions of KPS and KAS are very close to each other [4][8][6]. BESs are designed to enable secure broadcast transmissions and have been introduced in [13]. The broadcast encryption idea has grown in various directions: traitor tracing [11], anonymous broadcast transmission [16], re-keying protocols for secure multi-cast communications [9][10][22].

Our attention in this paper focuses on a model improving upon the weaknesses of a *single KDC*. Indeed, in the network model outlined before, a KDC must be *trusted*; moreover, it could become a communication *bottleneck* since all key request messages are sent to it and, last but not least, it could become a point of failure for the system: if the server crashes, secure communications cannot be supported anymore.

In [19] a new approach to key distribution was introduced to solve the above problems. A Distributed Key Distribution Center (DKDC, for short) is a set of n servers of a network that jointly realizes the same function of a Key Distribution Center. A user who needs to participate to a conference, sends a key-request to a subset at his choice of the n servers. The contacted servers answer with some information enabling the user to compute the conference key. In such a model, a single server by itself does not know the secret keys, since they are *shared* between the n servers, the communication bottleneck is eliminated, since the key-request messages are distributed, on average, along different paths, and

there is no single point of failure, since if a server crashes, the other are still able to support conference key computation.

In a subsequent paper [5], the notion of DKDC has been studied from an information theoretic point of view. Therein, the authors introduced the concept of a distributed key distribution scheme (DKDS, for short), a scheme realizing a DKDC, showing that the protocol proposed in [19], based on ℓ -wise independent functions, is optimal with respect to the amount of information needed to set up and manage the system.

In this paper we extend the model studied in [5], by considering a general family of subsets of servers, referred to as the *access structure*, that are authorized to help the users in recovering the conference keys. In the DKDSs studied in [5] the users must send a key-request to at least a certain number of servers, that is, only *threshold* access structures were considered. Because of the general framework, there will be possible to specify the access structure depending on the features of each one of the servers. We present bounds holding on the model using a reduction technique which relates DKDSs to Secret Sharing Schemes [3, 23]. This technique enables us to prove lower bounds on the memory storage, the communication complexity and the randomness needed to set up the scheme in an easy and elegant way. Moreover, we describe a linear algebraic approach to design DKDSs. Namely, we present a method to construct a DKDS from a linear secret sharing scheme and a family of linear ℓ -wise independent functions. The optimality of the obtained schemes relies on the optimality of the secret sharing schemes that are used in their construction. We show that some known previous constructions belong to the proposed framework. This approach is quite suitable since it allows a unified description of seemingly different schemes.

In the journal version of this paper we will describe two construction techniques to realize DKDS for any access structure. The first construction we will consider is based on the cumulative array method introduced in [25]. The second construction will be based on the technique of Benaloh and Leichter [2] which can be applied to any monotone formula describing the access structure.

Organization of the Paper. Some basic facts about secret sharing are recalled in Section 2. A Model for distributed key distribution schemes and the notation we use in the paper are given in Section 3. Some lower bounds on the amount of information that must be stored by the servers and on the size in bits of the messages each server has to send in order to reply key requests, are given in Section 4. There, it is also shown a lower bound on the randomness needed by the scheme. These bounds are found by using the relation between DKDSs secret sharing schemes and depend on the optimal value for the information rate of secret sharing schemes with access structure in the set of servers. In Section 5, we present a linear algebraic method to construct DKDSs from linear secret sharing schemes.

2 Secret Sharing Schemes

A secret sharing scheme (SSS, for short) is a protocol by means of which a dealer distributes a secret $k \in K$ into shares among a set of participants \mathcal{S} in such a way that only the authorized subsets of \mathcal{S} can reconstruct the secret k , whereas the participants in any non authorized subset of \mathcal{S} cannot determine anything about the value of the secret. Secret sharing were introduced in 1979 by Blakley [3] and Shamir [23]. The readers unfamiliar with secret sharing can find an excellent introduction to this topic in [26].

The authorized subsets form the *access structure* $\mathcal{A} \subset 2^{\mathcal{S}}$ of the SSS. The subsets belonging to \mathcal{A} are called *authorized* subsets, while those not in \mathcal{A} are called *forbidden*. We consider only *monotone* access structures, that is, any set containing an authorized subset must be authorized.

Using information theory, the two properties a secret sharing scheme must satisfy can be stated as follows: assuming that A denotes both a subset of participants and the set of shares these participants receive from the dealer to share a secret $k \in K$, and indicating the corresponding random variables in bold, it holds

- *Any authorized subset can compute the secret:* formally, for each $A \in \mathcal{A}$, $H(\mathbf{K}|\mathbf{A}) = 0$
- *Any forbidden subset has no information on the secret:* formally, for each $F \notin \mathcal{A}$, $H(\mathbf{K}|\mathbf{F}) = H(\mathbf{K})$

The efficiency of a secret sharing scheme is usually quantified by some measurements; basically the *information rate*, and the *randomness* needed to set up the scheme. The information rate $\rho(\Sigma, \mathcal{A}, K)$ of a secret sharing scheme Σ with access structure \mathcal{A} and set of secrets K is defined as the ratio between the *size of the secret* (measured in bits) and the *maximum size of the shares* given to the participants, while the randomness is the *number of random bits* used to setup the scheme. Secret sharing schemes with information rate $\rho = 1$, which is the maximum possible value of this parameter, are called *ideal*. An access structure \mathcal{A} on \mathcal{S} is said to be *ideal* if there exists an ideal secret sharing scheme with access structure \mathcal{A} .

Given an access structure \mathcal{A} on \mathcal{S} , we will indicate with $\rho^*(\mathcal{A})$ the optimal information rate for a SSS with access structure \mathcal{A} . More precisely, $\rho^*(\mathcal{A}) = \sup \rho(\Sigma, \mathcal{A}, K)$, where the supremum is taken over all possible sets of secrets K with $|K| \geq 2$ and all secret sharing schemes Σ with access structure \mathcal{A} .

We recall next some basic facts about linear secret sharing schemes. Let E be a vector space with finite dimension over the finite field $GF(q)$. For every $S_i \in \mathcal{S} \cup \{D\}$, where $D = S_0 \notin \mathcal{S}$ is a special participant called *dealer*, let us consider a vector space E_i over $GF(q)$ and a surjective linear mapping $\pi_i : E \rightarrow E_i$. Let us suppose that these linear mappings verify that, for any $A \subset \mathcal{S}$,

$$\bigcap_{S_i \in A} \ker \pi_i \subset \ker \pi_0 \quad \text{or} \quad \bigcap_{S_i \in A} \ker \pi_i + \ker \pi_0 = E.$$

This family of vector spaces and linear surjective mappings determines the access structure

$$\mathcal{A} = \left\{ A \subset \mathcal{S} : \bigcap_{S_i \in A} \ker \pi_i \subset \ker \pi_0 \right\}.$$

A secret sharing scheme with set of secrets $K = E_0$ and access structure \mathcal{A} is defined as follows: for a secret value $k \in E_0$, a vector $v \in E$ such that $\pi_0(v) = k$ is taken at random and every participant $S_i \in \mathcal{S}$ receives as its share the vector $a_i = \pi_i(v) \in E_i$. It is not difficult to prove that this is a secret sharing scheme with access structure \mathcal{A} . The information rate of this scheme is $\rho = \dim E_0 / (\max_{1 \leq i \leq n} \dim E_i)$. Secret sharing schemes constructed in this way are called *linear secret sharing schemes* (LSSSs for short). In a LSSS, the secret is computed by a linear mapping from the shares of the participants in an authorized subset. That is, for every $A = \{S_{i_1}, \dots, S_{i_r}\} \in \mathcal{A}$, there exists a linear mapping $\chi_A : E_{i_1} \times \dots \times E_{i_r} \rightarrow E_0$ that enables the participants in A to compute the secret.

Linear secret sharing schemes were first introduced by Brickell [7], who considered only ideal linear schemes with $\dim E_i = 1$ for any $S_i \in \mathcal{S} \cup \{D\}$. General linear secret sharing schemes were introduced by Simmons [24], Jackson and Martin [15] and Karchmer and Wigderson [17] under other names such as geometric secret sharing schemes or monotone span programs. In an ideal linear secret sharing scheme with $\dim E_0 = 1$, we can consider that the surjective linear mappings π_i are non-zero vectors in the dual space E^* . In that case, a subset $A \subset \mathcal{S}$ is authorized if and only if the vector $\pi_0 \in E^*$ can be expressed as a linear combination of the vectors $\{\pi_i \mid S_i \in A\}$. The access structures that can be defined in this way are called *vector space access structures*. Threshold access structures are a particular case of vector space access structures. Effectively, if \mathcal{A} is the (t, n) -threshold access structure, we can take $q > n$ a prime power and $x_i \in GF(q)$, for any $p_i \in \mathcal{S}$, non-zero pairwise different elements and consider $E = GF(q)^t$, $\pi_0 = (1, 0, \dots, 0) \in E^*$ and $\pi_i = (1, x_i, x_i^2, \dots, x_i^{t-1}) \in E^*$ for any $i = 1, \dots, n$. The ideal linear scheme we obtain in this way is in fact equivalent to the Shamir's threshold scheme [23].

Using the *monotone circuit construction* due to Ito, Saito and Nishizeki [14], Simmons, Jackson and Martin [25] proved that any access structure \mathcal{A} can be realized by a linear secret sharing scheme. The main drawback of the LSSSs that are constructed by using the general method proposed in [25] is that their information rates are in general very small.

Nevertheless, using decomposition techniques, linear secret sharing schemes with much better information rate can be found for some access structures. Those techniques consist of decomposing the given access structure \mathcal{A} into several substructures and combining secret sharing schemes on these substructures in order to obtain a secret sharing scheme for \mathcal{A} . For instance, one of the most powerful decomposition techniques to construct secret sharing schemes with good information rate is the *λ -decomposition construction* due to Stinson [27]. A linear secret sharing scheme is obtained when combining linear schemes in a λ -decomposition construction.

3 The Model

Let $\mathcal{U} = \{U_1, \dots, U_m\}$ be a set of m users and let $\mathcal{S} = \{S_1, \dots, S_n\}$ be a set of n servers. Each user has private connections with *all* the servers. Let us consider an access structure $\mathcal{A} \subset 2^{\mathcal{S}}$ on the set of servers and two families $\mathcal{C}, \mathcal{G} \subset 2^{\mathcal{U}}$ of subsets of the set of users. \mathcal{C} is the set of *conferences* and \mathcal{G} is the family of *tolerated coalitions*. A distributed key distribution scheme is divided in three phases: an *initialization phase*, which involves only the servers; a *key-request phase*, in which users ask for keys to servers; and a *key-computation phase*, in which users retrieve keys from the messages received from the servers contacted during the key request phase.

Initialization Phase. We assume that the initialization phase is performed by a *privileged* subset of servers $P_I = \{S_1, \dots, S_t\} \in \mathcal{A}$. Each of these servers, using a *private source* of randomness r_i , generates some information that securely distributes to the others. More precisely, for $i = 1, \dots, t$, S_i sends to S_j the value $\gamma_{i,j}$, where $j = 1, \dots, n$. At the end of the distribution, for $i = 1, \dots, n$, each server S_i *computes and stores* some secret information $a_i = f(\gamma_{1,i}, \dots, \gamma_{t,i})$, where f is a publicly known function.

Key-Request Phase. Let $C_h \in \mathcal{C}$ be a conference, that is, a group of users who need to securely communicate. Each user U_j in C_h , contacts the servers belonging to some subset $P \in \mathcal{A}$, requiring a key for the conference C_h . We denote such a key by κ_h . Server $S_i \in P$, contacted by user U_j , checks¹ for membership of U_j in C_h ; if the check is satisfied, he computes a value $y_{i,j}^h = F(a_i, j, h)$. Otherwise, he sets $y_{i,j}^h = \perp$, a special value which does convey no information about κ_h . Finally, S_i sends the value $y_{i,j}^h$ to U_j .

Key-Computation Phase. Once having received the answers from the contacted servers, each user U_j in C_h computes $\kappa_h = G_P(y_{i_1,j}^h, \dots, y_{i_{|P|},j}^h)$, with $i_1, \dots, i_{|P|}$ those indices of the contacted servers, and G_P is a publicly known function.

We are interested in formalizing, within an information theoretic framework², the notion of a DKDS, in order to quantify *exactly* the amount of resources that a *real-world* implementation of such a system can require. To this aim, we need to setup our notation.

- Let $\mathcal{C} \subset 2^{\mathcal{U}}$ be the set of conferences on \mathcal{U} indexed by elements of $\mathcal{H} = \{1, 2, \dots\}$.
- For any coalition $G = \{U_{j_1}, \dots, U_{j_g}\} \in \mathcal{G}$ of users, denote by $\mathcal{C}_G = \{C_h \in \mathcal{C} : C_h \cap G \neq \emptyset\}$ the set of conferences containing some user in G , and by $\mathcal{H}_G = \{h \in \mathcal{H} : C_h \in \mathcal{C}_G\}$ the set of corresponding indices. Let us

¹ We do not consider the underline authentication mechanism involved in a key request phase.

² The reader is referred to the Appendix A for the definition of the entropy function and some basic properties.

consider $\ell = \ell_G = \max_{G \in \mathcal{G}} |\mathcal{C}_G|$, the maximum number of conferences that are controlled by a coalition in \mathcal{G} .

- For $i = 1, \dots, t$, let $\Gamma_{i,j}$ be the set of values $\gamma_{i,j}$ that can be sent by server S_i to server S_j , for $j = 1, \dots, n$, and let $\Gamma_j = \Gamma_{1,j} \times \dots \times \Gamma_{t,j}$ be the set of values that S_j , for $j = 1, \dots, n$, can receive during the initialization phase.
- Let K_h be the set of possible values for κ_h , and let A_i be the set of values a_i the server S_i can compute during the initialization phase.
- Finally, let $Y_{i,j}^h$ be the set of values $y_{i,j}^h$ that can be sent by S_i when it receives a key-request message from U_j for the conference C_h .

Given three sets of indices $X = \{i_1, \dots, i_r\}$, $Y = \{j_1, \dots, j_s\}$, and $H = \{h_1, \dots, h_t\}$, and three families of sets $\{\Gamma_i\}$, $\{\Gamma_{i,j}\}$ and $\{Y_{i,j}^h\}$, we will denote by $T_X = T_{i_1} \times \dots \times T_{i_r}$, $T_{X,Y} = T_{i_1,j_1} \times \dots \times T_{i_r,j_s}$, and by $T_{X,Y}^H = T_{i_1,j_1}^{h_1} \times \dots \times T_{i_r,j_s}^{h_t}$, the corresponding Cartesian products. According to this notation, we will consider several Cartesian products, defined on the sets of our interest (see Table 1).

Table 1. Cartesian Products

Γ_Y	Set of values that can be received by server S_j , for $j \in Y$
$\Gamma_{X,j}$	Set of values that can be sent by server S_i to S_j , for $i \in X$
$\Gamma_{X,Y}$	Set of values that can be sent by server S_i to S_j , for $i \in X$ and $j \in Y$
K_X	Set of $ X $ -tuple of conference keys
A_X	Set of $ X $ -tuple of private information a_i
$Y_{X,j}^h$	Set of values that can be sent by S_i , for $i \in X$, to U_j for the conference C_h
Y_G^h	Set of values that can be sent by S_1, \dots, S_n to U_j , with $j \in G$, for C_h
Y_G^H	Set of values that can be sent by S_1, \dots, S_n to U_j , with $j \in G$, for $C_h \forall h \in H$

We will denote in boldface the random variables $\mathbf{\Gamma}_{i,j}, \mathbf{\Gamma}_j, \dots, \mathbf{Y}_G^X$ assuming values on the sets $\Gamma_{i,j}, \Gamma_j, \dots, Y_G^X$, according to the probability distributions $\mathcal{P}_{\mathbf{\Gamma}_{i,j}}, \mathcal{P}_{\mathbf{\Gamma}_j}, \dots, \mathcal{P}_{\mathbf{Y}_G^X}$.

Roughly speaking, a DKDC must satisfy the following properties:

- **Correct Initialization Phase.** When the initialization phase correctly terminates, each server S_i must be able to compute his private information a_i . On the other hand, if server S_i misses/does-not-receive *just one* message from the servers in P_I sending information, then S_i must not gain any information about a_i . We model these two properties by relations 1 and 2 of the formal definition.
- **Consistent Key Computation.** Each user in a conference $C_h \subseteq \mathcal{U}$ must be able to compute *the same* conference key, after interacting with the servers

³ Without loss of generality, we choose P_I as one of the smallest subsets in \mathcal{A} because one of our aim is to minimize the randomness and the communication complexity of the initialization phase.

of a subset $P \in \mathcal{A}$ at his choice. Relations 3 and 4 of the formal definition ensure these properties. More precisely, relation 3 establishes that each server uniquely determines an answer to any key-request message; while, property 4 establishes that each user uniquely computes the same conference key, using the messages received by the subset of authorized servers he has contacted for that conference key.

- **Conference Key Security.** A conference key must be secure against attacks performed by coalitions of servers, coalitions of users, and hybrid coalitions (servers and users). This is the most intriguing and difficult property to formalize. Indeed, the worst case scenario to look after consists of a coalition of users $G \in \mathcal{G}$ that honestly run the protocol many times, retrieving several conference keys and, then, with the cooperation of some dishonest servers, try to gain information on a new conference key, they didn't ask before. Notice that, according to our notation, the maximum amount of information the coalition can acquire honestly running the protocol is represented by $\mathbf{Y}_G^{\mathcal{H}_G \setminus \{h\}}$; moreover, dishonest servers, belonging to $F \notin \mathcal{A}$, know $\mathbf{\Gamma}_F$ and, maybe, $\mathbf{\Gamma}_{Z,N}$. This random variable takes into account the possibility that some of the dishonest servers send information in the initialization phase (i.e. $Z \subseteq F \cap P_I$). Hence, they know the messages they send out to the other servers in this phase. Relation 5 ensures that such coalitions of adversaries, do not gain information on any new key.

Formally, a Distributed Key Distribution Scheme with access structure \mathcal{A} on \mathcal{S} can be defined as follows:

Definition 1. Let $\mathcal{U} = \{U_1, \dots, U_m\}$ be a set of users and let $\mathcal{S} = \{S_1, \dots, S_n\}$ be a set of servers. Let us consider an access structure $\mathcal{A} \subset 2^{\mathcal{S}}$ on the set of servers and two families $\mathcal{C}, \mathcal{G} \subset 2^{\mathcal{U}}$ of subsets of the set of users. An $(\mathcal{A}, \mathcal{C}, \mathcal{G})$ -Distributed Key Distribution Scheme (for short, $(\mathcal{A}, \mathcal{C}, \mathcal{G})$ -DKDS) is a protocol which enables each user of $C_h \in \mathcal{C}$ to compute a common key κ_h interacting with a subset of authorized servers in \mathcal{A} of the network. More precisely, the following properties are satisfied:

- 1 For each $i = 1, \dots, n$, $H(\mathbf{A}_i | \mathbf{\Gamma}_i) = 0$.
- 2 For each $X \subset P_I$, $X \neq P_I$, and $i \in \{1, \dots, n\}$, $H(\mathbf{A}_i | \mathbf{\Gamma}_{X,i}) = H(\mathbf{A}_i)$.
- 3 For each $C_h \in \mathcal{C}$, for each $U_j \in C_h$, and for each $i = 1, \dots, n$, $H(\mathbf{Y}_{i,j}^h | \mathbf{A}_i) = 0$.
- 4 For each $C_h \in \mathcal{C}$, for each $P \in \mathcal{A}$, and for each $U_j \in C_h$, $H(\mathbf{K}_h | \mathbf{Y}_{P,j}^h) = 0$.
- 5 For each $C_h \in \mathcal{C}$, for each $G \in \mathcal{G}$, and for each subset $F \notin \mathcal{A}$

$$H(\mathbf{K}_h | \mathbf{Y}_G^{\mathcal{H}_G \setminus \{h\}} \mathbf{\Gamma}_F \mathbf{\Gamma}_{Z,N}) = H(\mathbf{K}_h)$$

where $Z = F \cap P_I$ and $N = \{1, \dots, n\}$.

Notice that a DKDC implemented by a DKDS is a *deterministic* system at all. Random bits are needed only at the beginning (i.e. initialization of the system),

when each server in P_I uses his own random source to generate messages to deliver to the other servers of the network.

In the following, without loss of generality and to emphasize the *real-world oriented* motivations of our study, we assume that the conference keys are *uniformly chosen* in a set K . Hence, for different $h, h' \in \mathcal{H}$, $H(\mathbf{K}_h) = H(\mathbf{K}_{h'}) = \log |K|$. Moreover, since from the Appendix A follows that $H(\mathbf{X}) \leq \log |X|$, for each random variable \mathbf{X} assuming values on the set X , we will enunciate the lower bounds in terms of the size of the sets of our interest.

4 Lower Bounds

A basic relation between $(\mathcal{A}, \mathcal{C}, \mathcal{G})$ -DKDS and Secret Sharing Schemes enables us to derive some lower bounds on the *memory storage*, on the *communication complexity*, and on the number of *random bits* needed to set up the scheme.

We state some results which will be useful in proving the lower bounds. Due to space limitation, we omit the proofs, but the reader can easily convince himself that the results hold (the formal proofs is just a matter of simple algebra and basic information theory inequalities).

The following simple lemma establishes that, given three random variables \mathbf{A} , \mathbf{B} , and \mathbf{C} , if \mathbf{B} is a function of \mathbf{C} , then \mathbf{B} gives less information on \mathbf{A} than \mathbf{C} .

Lemma 1. *Let \mathbf{A} , \mathbf{B} , and \mathbf{C} be three random variables such that $H(\mathbf{B}|\mathbf{C}) = 0$. Then, $H(\mathbf{A}|\mathbf{B}) \geq H(\mathbf{A}|\mathbf{C})$.*

The next lemma, instead, establishes that the amount of information a subset of servers gains about the conference keys depends on the *membership* of the subset along the access structure \mathcal{A} , and is *all-or-nothing* in fashion.

Lemma 2. *Let P and F be two subsets of \mathcal{S} such that $P \in \mathcal{A}$ and $F \notin \mathcal{A}$. Moreover, let $\mathcal{H}_r = \{h_1 \dots, h_r\} \subseteq \mathcal{H}$ be a subset of indices of conferences. Then, it holds that*

$$H(\mathbf{K}_{\mathcal{H}_r}|\mathbf{A}_P) = 0, \text{ while } H(\mathbf{K}_{\mathcal{H}_r}|\mathbf{A}_F) = H(\mathbf{K}_{\mathcal{H}_r}).$$

Finally, the conference keys a coalition of users can retrieve are statistically independent.

Lemma 3. *Let $G = \{U_{j_1}, \dots, U_{j_g}\} \subseteq \mathcal{U}$ be a coalition of users, and let $\mathcal{H}_G = \{h_1, \dots, h_{\ell_G}\}$. Then, for each $r = 1, \dots, \ell_G$, it holds that*

$$H(\mathbf{K}_{h_r}|\mathbf{K}_{\mathcal{H}_G \setminus \{h_r\}}) = H(\mathbf{K}_{h_r}).$$

Lower bounds on the amount of information each server has to store and send to a key-request message, and on the number of random bits needed to set up the scheme can be established exploring the aforementioned relation existing between a DKDS and SSSs. First of all, notice that, the 4-th and the 5-th conditions of the definition of a DKDS “contain” a SSS. More precisely, in any DKDS

- for each $C_h \in \mathcal{C}$, for each $P \in \mathcal{A}$, and for each $U_j \in C_h$, $H(\mathbf{K}_h | \mathbf{Y}_{P,j}^h) = 0$
- for each $C_h \in \mathcal{C}$, for each $G = \{U_{j_1}, \dots, U_{j_g}\}$, and for each $F \notin \mathcal{A}$ it holds that

$$H(\mathbf{K}_h | \mathbf{Y}_G^{\mathcal{H}_G \setminus \{h\}} \mathbf{T}_F \mathbf{T}_{Z,N}) = H(\mathbf{K}_h)$$

The first relation is exactly the *reconstruction property* of a SSS Σ_1 with access structure \mathcal{A} and set of secrets K_h , shared among the servers with shares $y_{1,j}^h, \dots, y_{n,j}^h$. The second one *contains the security condition*. Indeed, since the values $y_{i,j}^h$ are function of the private information a_i computed and stored by each server at the end of the initialization phase, using equation (6) in Appendix A, it is easy to check that

$$H(\mathbf{K}_h) = H(\mathbf{K}_h | \mathbf{Y}_G^{\mathcal{H}_G \setminus \{h\}} \mathbf{T}_F \mathbf{T}_{Z,N}) \leq H(\mathbf{K}_h | \mathbf{A}_F) \leq H(\mathbf{K}_h | \mathbf{Y}_{F,j}^h) \leq H(\mathbf{K}_h).$$

Therefore, the size in bits of each answer that is sent by a server to a key request message must satisfy the inequality given by the following theorem:

Theorem 1. *In any $(\mathcal{A}, \mathcal{C}, \mathcal{G})$ -DKDS the amount of information each server has to send to a key request message must be, for all $j = 1, \dots, m$ and for each $h \in \mathcal{H}$, such that*

$$\max_{i=1, \dots, n} \log |Y_{i,j}^h| \geq \frac{\log |K|}{\rho^*},$$

where ρ^* is the optimal information rate for a SSS with access structure \mathcal{A} .

Analogously, we can show a lower bound on the amount of information each server has to store. To this aim, notice that each server holds a *share of the sequence of keys* the users can ask for. If we denote by ℓ the maximum number of conference keys a coalition of users can retrieve, then the servers actually share a secret which can be seen as an element belonging to the set $T = K_{\mathcal{H}_\ell}$, where $\mathcal{H}_\ell = \{h_1, \dots, h_\ell\}$. Furthermore, Lemma 3 implies that

$$H(\mathbf{T}) = H(\mathbf{K}_{\mathcal{H}_\ell}) = \sum_j H(\mathbf{K}_{i_j}) = \ell H(\mathbf{K}).$$

Since Lemma 2 establishes that $H(\mathbf{T} | \mathbf{A}_P) = 0$ if $P \in \mathcal{A}$, while $H(\mathbf{T} | \mathbf{A}_F) = H(\mathbf{T})$, when $F \notin \mathcal{A}$, we recover another SSS, say $(\Sigma_2, \text{with access structure } \mathcal{A} \text{ and set of secrets } T)$. Consequently, the next theorem holds:

Theorem 2. *In any $(\mathcal{A}, \mathcal{C}, \mathcal{G})$ -DKDS the amount of information server S_i has to store must be*

$$\max \log |A_i| \geq \frac{\log |T|}{\rho^*} = \ell \frac{\log |K|}{\rho^*},$$

where ρ^* is the optimal information rate for a SSS with access structure \mathcal{A} .

The communication complexity of a $(\mathcal{A}, \mathcal{C}, \mathcal{G})$ -DKDS can be lower bounded as follows: notice that relations 1 and 2 of a DKDS are again the properties

characterizing a secret sharing scheme, say Σ_3 . More precisely, for any subset $F \subset P_I$, it holds that

$$H(\mathbf{A}_i | \Gamma_i) = 0, \text{ while } H(\mathbf{A}_i | \Gamma_{F,i}) = H(\mathbf{A}_i).$$

In this case $\mathcal{S} = \{S_1, \dots, S_t\}$ is the *only* subset in the access structure \mathcal{A}' of the SSS Σ_3 (i.e., a (t, t) threshold structure), and the shared secret is exactly a_i . Hence, the following holds:

Theorem 3. *In any $(\mathcal{A}, \mathcal{C}, \mathcal{G})$ -DKDS, for $j = 1, \dots, t$, the amount of information each S_j has to send during the initialization phase must be*

$$\log |\Gamma_{j,i}| \geq \log |A_i|.$$

Moreover, since each server performing the initialization phase uses a private source of random bits, we have:

Theorem 4. *In any $(\mathcal{A}, \mathcal{C}, \mathcal{G})$ -DKDS the amount of information each server receives during the initialization phase is*

$$\log |\Gamma_j| = \log |\Gamma_{1,j}| \times \dots \times \log |\Gamma_{t,j}| = \sum_{i=1}^t \log |\Gamma_{i,j}|.$$

Finally, a lower bound on the number of random bits, referred to as the *randomness* of the scheme and denoted by \mathcal{R} , needed to set up the scheme can be easily obtained using the above results.

Theorem 5. *In any $(\mathcal{A}, \mathcal{C}, \mathcal{G})$ -DKDS the randomness satisfies*

$$\mathcal{R} \geq t\ell R_{opt}$$

where R_{opt} is the minimum amount of randomness required by a secret sharing scheme Σ with access structure \mathcal{A} and set of secrets K .

Hence, all the results and bounds on SSS concerning randomness and information rates related to the study of specific access structures can be used to retrieve corresponding results and bounds holding for $(\mathcal{A}, \mathcal{C}, \mathcal{G})$ -DKDSs.

5 Protocols: Designing DKDSs from LSSSs

In this section, we present a method to construct a $(\mathcal{A}, \mathcal{C}, \mathcal{G})$ -DKDS from any linear secret sharing scheme with access structure \mathcal{A} . Since there exists a LSSS realizing any access structure \mathcal{A} , our method will provide DKDSs for any general access structure. The optimality of the $(\mathcal{A}, \mathcal{C}, \mathcal{G})$ -DKDS relies upon the optimality of the LSSS, that is, if we can find a LSSS with optimal information rate for the access structure \mathcal{A} , we will be able to construct an optimal $(\mathcal{A}, \mathcal{C}, \mathcal{G})$ -DKDS.

Let \mathcal{A} be an access structure on the set of servers $\mathcal{S} = \{S_1, S_2, \dots, S_n\}$. Let \mathcal{C} and \mathcal{G} be, respectively, the families of conferences and tolerated coalitions of

users. Let ℓ be the maximum number of conferences that are controlled by a coalition in \mathcal{G} .

Let E, E_0, E_1, \dots, E_n be vector spaces of finite dimension over a finite field $GF(q)$ such that there exist surjective linear mappings $\pi_0 : E \rightarrow E_0$ and $\pi_i : E \rightarrow E_i$, where $i = 1, \dots, n$, defining a LSSS Σ on \mathcal{S} with access structure \mathcal{A} . Let us recall that, for every authorized subset of servers $A = \{S_{i_1}, \dots, S_{i_r}\} \in \mathcal{A}$, there exists a linear mapping $\chi_A : E_{i_1} \times \dots \times E_{i_r} \rightarrow E_0$ that makes it possible to compute the secret from the shares. For any $i = 0, 1, \dots, n$, we consider the linear mapping $\pi_i^\ell : E^\ell \rightarrow E_i^\ell$ defined by $\pi_i^\ell(u_1, \dots, u_\ell) = (\pi_i(u_1), \dots, \pi_i(u_\ell))$. It is not difficult to check that the mappings π_i^ℓ define a LSSS Σ^ℓ with set of secrets E_0^ℓ and the same access structure and information rate as the LSSS Σ . In this case, the secret is computed from the shares by using the linear mappings $\chi_A^\ell : E_{i_1}^\ell \times \dots \times E_{i_r}^\ell \rightarrow E_0^\ell$.

Let us consider linear forms $\varphi_h : GF(q)^\ell \rightarrow GF(q)$, where $h = 1, \dots, |\mathcal{C}|$, such that any different ℓ forms $\varphi_{h_1}, \dots, \varphi_{h_\ell}$ are linearly independent. Observe that a form φ_h is a vector in the dual space $(GF(q)^\ell)^*$ and is determined by its coordinates $(\lambda_{h,1}, \dots, \lambda_{h,\ell})$, where $\varphi_h(v_1, \dots, v_\ell) = \sum_{j=1}^\ell \lambda_{h,j} v_j$. If $q \geq |\mathcal{C}|$, such a family of linear forms can be constructed by considering $|\mathcal{C}|$ different values $z_1, \dots, z_{|\mathcal{C}|}$ in the finite field $GF(q)$ and taking the vectors $(\lambda_{h,1}, \dots, \lambda_{h,\ell}) = (1, z_h, z_h^2, \dots, z_h^{\ell-1})$, where $h = 1, \dots, |\mathcal{C}|$. These linear forms define a *linear key generator* \mathcal{K} , that can be used to determine the conference keys κ_h in the following way: a vector $v \in GF(q)^\ell$ is chosen uniformly at random and $\kappa_h = \varphi_h(v)$ is the key for the conference C_h for every $C_h \in \mathcal{C}$. In this way, the knowledge of the conference keys $\kappa_{h_1}, \dots, \kappa_{h_{\ell-1}}$ does not provide any information on the value of κ_{h_ℓ} . For any vector space U , we can extend the linear key generator \mathcal{K} , which provides conference keys that are in the finite field $GF(q)$, to a linear key generator \mathcal{K}^U whose keys κ_h are vectors in U . This can be done by considering the linear mappings $\varphi_h^U : U^\ell \rightarrow U$ defined by $\varphi_h^U(u_1, \dots, u_\ell) = \sum_{j=1}^\ell \lambda_{h,j} u_j$, where $(\lambda_{h,1}, \dots, \lambda_{h,\ell})$ are the coordinates of the linear form φ_h . As before, any ℓ different conference keys are independent.

We describe next how to construct a $(\mathcal{A}, \mathcal{C}, \mathcal{G})$ -DKDS from the LSSS Σ and the linear key generator \mathcal{K} . Let us consider the LSSS Σ^ℓ , the linear key generator \mathcal{K}^E and, for any $i = 0, 1, \dots, n$, the linear key generator $\mathcal{K}^i = \mathcal{K}^{E_i}$, which consists of the linear mappings $\varphi_h^i = \varphi_h^{E_i}$. Observe that $\varphi_h^i \circ \pi_i^\ell = \pi_i \circ \varphi_h^E$ for any $C_h \in \mathcal{C}$ and $i = 0, 1, \dots, n$. In effect,

$$\begin{aligned} (\varphi_h^i \circ \pi_i^\ell)(u) &= \varphi_h^i(\pi_i(u_1), \dots, \pi_i(u_\ell)) = \sum_{j=1}^\ell \lambda_{h,j} \pi_i(u_j) = \pi_i \left(\sum_{j=1}^\ell \lambda_{h,j} u_j \right) = \\ &= (\pi_i \circ \varphi_h^E)(u) \text{ for any } u = (u_1, \dots, u_\ell) \in E^\ell. \end{aligned}$$

Initialization Phase. This phase is carried out by an authorized subset of servers $P_I \in \mathcal{A}$. We can suppose that $P_I = \{S_1, \dots, S_t\}$.

- For every $i = 1, \dots, t$, the server S_i chooses at random a vector $r_i \in E^\ell$ and, for every $j = 1, \dots, n$, sends to server S_j the vector $\pi_j^\ell(r_i) \in E_j^\ell$.
- For $j = 1, \dots, n$, each server S_j computes his private information summing up the shares it has received from the servers in P_I . That is, server S_j computes $a_j = \pi_j^\ell(r_1) + \dots + \pi_j^\ell(r_t) = \pi_j^\ell(u) \in E_j^\ell$, where $u = r_1 + \dots + r_t \in E^\ell$.

Therefore, each server S_i has a vector $a_i = (a_{i1}, \dots, a_{i\ell}) \in E_i^\ell$ after the initialization phase. This vector is in fact a share of a secret vector $\pi_0^\ell(u) = v = (v_1, \dots, v_\ell) \in E_0^\ell$ by the LSSS Σ^ℓ . By definition, the key corresponding to the conference $C_h \in \mathcal{C}$ is $\kappa_h = (\varphi_h^0 \circ \pi_0^\ell)(u) \in E_0$.

Key Request Phase. A user in a conference C_h who wants to obtain the conference key κ_h , sends a key-request message for the conference key to an authorized subset of servers $A = \{S_{i_1}, \dots, S_{i_r}\} \in \mathcal{A}$. Each server S_i invoked by the user checks that the user belongs to C_h and sends to the user the vector $\varphi_h^i(a_i) = (\varphi_h^i \circ \pi_i^\ell)(u) = (\pi_i \circ \varphi_h^E)(u) \in E_i$, which is indeed a share of the conference key $\kappa_h = (\varphi_h^0 \circ \pi_0^\ell)(u) = (\pi_0 \circ \varphi_h^E)(u) \in E_0$ by the LSSS Σ .

Key Computation Phase. Using the values received from the servers in $A \in \mathcal{A}$ the user in C_h recovers the secret key by computing $\kappa_h = \chi_A(\varphi_h^{i_1}(a_{i_1}), \dots, \varphi_h^{i_r}(a_{i_r}))$.

One has to check that the proposed scheme verifies properties 1–5 in Definition 1. This is done by applying the properties of the linear secret sharing scheme Σ and the linear key generator \mathcal{K} .

Finally, we compare the parameters in our scheme with the bounds given in Section 4. Let

$$\rho = \frac{\dim E_0}{\max_{1 \leq i \leq n} \dim E_i}$$

be the information rate of the LSSS Σ . Let q (a power of a prime) be the cardinality of the finite field $GF(q)$.

The amount of information that a server $S_i \in \mathcal{S}$ has to send to a user $U_j \in C_h$ in the key request phase is $\log |Y_{i,j}^h| = \log |E_i| = \log q \dim E_i$. Observe that

$$\max_{i=1, \dots, n} \log |Y_{i,j}^h| = \log q \dim E_0 \frac{\max_{i=1, \dots, n} \dim E_i}{\dim E_0} = \frac{\log |K|}{\rho}.$$

Therefore, the bound given in Theorem 1 is attained if Σ has optimal information rate, that is, if $\rho = \rho^*(\mathcal{A})$. Equally, it is not difficult to check that, in this case, the bound given in Theorem 2 is also attained. We observe that the randomness that is needed to set up the scheme is $|P_I| \log |E^\ell| = t\ell \log q \dim E$. Since the randomness of the LSSS Σ is equal to $\log q \dim E$, we see that the bound in Theorem 3 is also attained whenever the randomness of Σ is optimal.

Remark 1. The *linearity* property of the secret sharing scheme is *not* necessary to design a DKDS. Actually, from *any* secret sharing scheme, realizing a given access structure, we can set up a DKDS on the same access structure. The reader can easily convince himself noticing that in our protocol each server sums up the shares obtained during the distribution phase. This is one of the steps in which the linearity property of the scheme is applied. If the secret sharing scheme is not linear, each server has to store *all* the shares received from the servers performing the distribution. Therefore, the amount of information that must be stored by each server is much larger when using a general secret sharing scheme instead of a linear one. On the other hand, when a user asks a server for a conference

key, he receives *several* shares that must be processed in order to recover the conference key. That is, the communication complexity is also much larger if the scheme is not constructed from a LSSS.

More precisely, let S_1, \dots, S_k be the servers performing the distribution. For each conference $C \in \mathcal{C}$, let the key κ_c be defined as $\kappa_c = \kappa_{c,1} + \dots + \kappa_{c,k} \in Z_q$, where $\kappa_{c,j} \in Z_q$ is a *random* value for each $j = 1, \dots, k$. Assuming that a secret sharing scheme for the given access structure \mathcal{A} exists for each $v \in Z_q$, a DKDS can be set up as follows: for $i = 1, \dots, k$, server S_i generates and shares $\kappa_{c,i}$ according to a secret sharing scheme for \mathcal{A} . For $j = 1, \dots, n$, server S_j receives $share_{c,1}^j, \dots, share_{c,k}^j$. A user who needs to recover the conference key κ_c , asks a subset of servers $\{S_{i_1}, \dots, S_{i_k}\} \in \mathcal{A}$ and receives $(share_{c,1}^{i_1}, \dots, share_{c,k}^{i_1}), \dots, (share_{c,1}^{i_k}, \dots, share_{c,k}^{i_k})$. From the sets of shares, he can compute $\kappa_{c,i}$, for $i = 1, \dots, k$. Therefore, he can compute κ_c . Thus, the linearity property is useful just for *efficiency reasons*.

Some Examples

A $(k, n, \mathcal{C}, \mathcal{G})$ -DKDS is a $(\mathcal{A}, \mathcal{C}, \mathcal{G})$ -DKDS where \mathcal{A} is a *threshold access structure* on the set $\mathcal{S} = \{S_1, \dots, S_n\}$ of n servers, that is, $\mathcal{A} = \{P \subset \mathcal{S} : |P| \geq k\}$. A construction of a (k, n, \mathcal{C}) -DKDS based on a family of ℓ -wise independent functions has been proposed in [19]. Subsequently, in [5], the construction has been proved to be optimal respect to the information distributed to the parties. Notice that a $(k, n, \mathcal{C}, \mathcal{G})$ -DKDS can be obtained as a particular case of our general construction. We only have to consider a LSSS Σ with threshold access structure, for instance, the Shamir's scheme [23], which has information rate $\rho = 1$. We obtain in this way an optimal $(k, n, \mathcal{C}, \mathcal{G})$ -DKDS, which is in fact equivalent to the one proposed in [19].

A $(\mathcal{A}, \mathcal{C}, \mathcal{G})$ -DKDS with the same values for the considered parameters (memory storage, communication complexity and randomness) can be obtained for any vector space access structure \mathcal{A} . In this case, we apply our construction to an ideal LSSS Σ with access structure \mathcal{A} .

Optimal $(\mathcal{A}, \mathcal{C}, \mathcal{G})$ -DKDSs can also be constructed for any access structure \mathcal{A} such that a LSSS with optimal information rate is known for \mathcal{A} . For instance, let us consider the access structure on a set $\mathcal{S} = \{S_1, S_2, S_3, S_4\}$ of 4 servers whose minimal authorized subsets are $\mathcal{A}_0 = \{\{S_1, S_2\}, \{S_2, S_3\}, \{S_3, S_4\}\}$. This access structure is well known in the literature concerning secret sharing schemes [8]. It has been proved in [8] that the information rate of any SSS for this access structure is at most $2/3$. Besides, there exists a linear secret sharing scheme Σ with information rate $\rho = 2/3$. That is, the optimal information rate of this access structure is $\rho^*(\mathcal{A}) = 2/3$ and there exists a LSSS for \mathcal{A} with $\rho = \rho^*$. Therefore, we can construct a $(\mathcal{A}, \mathcal{C}, \mathcal{G})$ -DKDS attaining the bounds in Section 4.

6 Conclusion and Open Problems

In this paper we have shown bounds and constructions for unconditionally secure DKDSs with a general access structure on the set of servers. Such schemes enable

to setup distributed KDCs which solve many problems related to the presence across a network of a single on-line KDC. Two main contributions can be found in this paper: the reduction technique applied to find the lower bounds and the linear algebraic framework which unifies many previous proposals. We emphasize the optimality of our construction. An interesting open problem is to consider the model where servers can be corrupted. Some verifiable DKDSs must be designed in order to guarantee the security of the scheme.

References

1. M. Bellare and P. Rogaway. Provably Secure Session Key Distribution: The Three Party Case. Proc. 27th Annual Symposium on the Theory of Computing, ACM, 1995.
2. J. Benaloh and J. Leichter, Generalized Secret Sharing and Monotone Functions. Lecture Notes in Comput. Sci., 403, 27–35, 1990.
3. G.R. Blakley. Safeguarding Cryptographic Keys. Proceedings of AFIPS 1979 National Computer Conference, Vol. 48, pp. 313–317, 1979.
4. R. Blom. An Optimal Class of Symmetric Key Generation Systems. Advances in Cryptology - Eurocrypt'84, Lecture Notes in Comput. Sci., vol. 209, pp. 335–338, 1984.
5. C. Blundo, and P. D'Arco. Unconditionally Secure Distributed Key Distribution Schemes. Available at <http://www.dia.unisa.it/paodar.dir>
6. C. Blundo, A. De Santis, A. Herzberg, S. Kutten, U. Vaccaro and M. Yung. Perfectly-Secure Key Distribution for Dynamic Conferences. Information and Computation, vol. 146, no. 1, pp. 1–23, 1998.
7. E.F. Brickell. Some ideal secret sharing schemes. J. Combin. Math. and Combin. Comput., 9, 105–113, 1989.
8. R.M. Capocelli, A. De Santis, L. Gargano and U. Vaccaro. On the Size of the Shares in Secret Sharing Schemes. Advances in cryptology - CRYPTO'91, Lecture Notes in Comput. Sci., 576, 101–113, 1992.
9. R. Canetti, J. Garey, G. Itkins, D. Micciaccio, M. Naor and B. Pinkas. Issues in Multicast Security: A Taxonomy and Efficient Constructions. Proceedings of INFOCOM '99, vol. 2, pp. 708–716, 1999.
10. R. Canetti, T. Malkin and K. Nissim. Efficient Communication-Storage Tradeoffs for Multicast Encryption. Advances in Cryptology - Eurocrypt'99, Lecture Notes in Comput. Sci., vol. 1592, pp. 459–474, 1999.
11. B. Chor, A. Fiat, and M. Naor. Tracing Traitors. Advances in Cryptology - Eurocrypt'94, Lecture Notes in Comput. Sci., vol. 950 pp. 257–270, 1994.
12. T.M. Cover and J.A. Thomas. Elements of Information Theory. John Wiley & Sons, 1991.
13. A. Fiat and M. Naor. Broadcast Encryption. Advances in Cryptology - Crypto 92, Lecture Notes in Comput. Sci., vol. 773, pp. 480–491, 1993.
14. M. Ito, A. Saito and T. Nishizeki. Secret sharing scheme realizing any access structure. Proc. IEEE Globecom'87, 99–102, 1987.
15. W. Jackson and K. Martin. Geometric Secret Sharing Schemes and Their Duals. Des. Codes Cryptogr., 4, 83–95, 1994.
16. M. Just, E. Kranakis, D. Krizanc, P. Van Oorschot. Key Distribution via True Broadcasting. Proceedings of the 2nd ACM Conference on Computer and Communications Security, pp. 81–88, 1994.

17. M. Karchmer, A. Wigderson. On span programs. Proc. of Structure in Complexity'93, 102–111, 1993.
18. T. Matsumoto and H. Imai. On the Key Predistribution System: A Practical Solution to the Key Distribution Problem. Advances in Cryptology - Eurocrypt'87, Lecture Notes in Comput. Science, vol. 239, pp. 185–193, 1987.
19. M. Naor, B. Pinkas, and O. Reingold. Distributed Pseudo-random Functions and KDCs. Advances in Cryptology - Eurocrypt'99, Lecture Notes in Comput. Sci., vol. 1592, pp. 327–346, 1999.
20. R. M. Needham and M. D. Schroeder. Using Encryption for Authentication in Large Networks of Computers. Communications of ACM, vol. 21, pp. 993–999, 1978.
21. B. C. Neuman and T. Tso. Kerberos: An Authentication Service for Computer Networks. IEEE Transactions on Communications, vol. 32, pp. 33–38, 1994.
22. R. Poovendran, J.S.Baras. An Information Theoretic Approach for Design and Analysis of Rooted-Tree Based Multicast Key Management Schemes. Advances in Cryptology - Crypto'99, Lecture Notes in Comput. Sci., vol. 1666, pp. 624–638, 1999.
23. A. Shamir. How to Share a Secret. Communications of ACM, vol. 22, n. 11, pp. 612–613, 1979.
24. G.J. Simmons. How to (really) share a secret. Advances in Cryptology, CRYPTO 88, Lecture Notes in Comput. Sci., 403, 390–448, 1990.
25. G.J. Simmons, W. Jackson and K. Martin. The geometry of secret sharing schemes. Bull. of the ICA, 1, 71–88, 1991.
26. D.R. Stinson. An explication of secret sharing schemes. Des. Codes Cryptogr., 2, 357–390, 1992.
27. D.R. Stinson. Decomposition Constructions for Secret-Sharing Schemes. IEEE Trans. on Information Theory, 40, 118–125, 1994.
28. D. R. Stinson. On Some Methods for Unconditional Secure Key Distribution and Broadcast Encryption. Designs, Codes and Cryptography, vol. 12, pp. 215–243, 1997.

A Information Theory Elements

This appendix briefly recalls some elements of information theory (see [12] for details). Let \mathbf{X} be a random variable taking values on a set X according to a probability distribution $\{P_{\mathbf{X}}(x)\}_{x \in X}$. The *entropy* of \mathbf{X} , denoted by $H(\mathbf{X})$, is defined as

$$H(\mathbf{X}) = - \sum_{x \in X} P_{\mathbf{X}}(x) \log P_{\mathbf{X}}(x),$$

where the logarithm is to the base 2. The entropy satisfies $0 \leq H(\mathbf{X}) \leq \log |X|$, where $H(\mathbf{X}) = 0$ if and only if there exists $x_0 \in X$ such that $Pr(\mathbf{X} = x_0) = 1$; whereas, $H(\mathbf{X}) = \log |X|$ if and only if $Pr(\mathbf{X} = x) = 1/|X|$, for all $x \in X$. Given two random variables \mathbf{X} and \mathbf{Y} taking values on sets X and Y , respectively, according to the joint probability distribution $\{P_{\mathbf{XY}}(x, y)\}_{x \in X, y \in Y}$ on their Cartesian product, the *conditional entropy* $H(\mathbf{X}|\mathbf{Y})$ is defined as

$$H(\mathbf{X}|\mathbf{Y}) = - \sum_{y \in Y} \sum_{x \in X} P_{\mathbf{Y}}(y) P_{\mathbf{X}|\mathbf{Y}}(x|y) \log P_{\mathbf{X}|\mathbf{Y}}(x|y).$$

It is easy to see that

$$H(\mathbf{X}|\mathbf{Y}) \geq 0. \quad (1)$$

with equality if and only if X is a function of Y . Given $n + 1$ random variables, $\mathbf{X}_1 \dots \mathbf{X}_n \mathbf{Y}$, the entropy of $\mathbf{X}_1 \dots \mathbf{X}_n$ given \mathbf{Y} can be written as

$$H(\mathbf{X}_1 \dots \mathbf{X}_n | \mathbf{Y}) = H(\mathbf{X}_1 | \mathbf{Y}) + H(\mathbf{X}_2 | \mathbf{X}_1 \mathbf{Y}) + \dots + H(\mathbf{X}_n | \mathbf{X}_1 \dots \mathbf{X}_{n-1} \mathbf{Y}). \quad (2)$$

The *mutual information* between \mathbf{X} and \mathbf{Y} is given by

$$I(\mathbf{X}; \mathbf{Y}) = H(\mathbf{X}) - H(\mathbf{X} | \mathbf{Y}).$$

Since, $I(\mathbf{X}; \mathbf{Y}) = I(\mathbf{Y}; \mathbf{X})$ and $I(\mathbf{X}; \mathbf{Y}) \geq 0$, it is easy to see that

$$H(\mathbf{X}) \geq H(\mathbf{X} | \mathbf{Y}), \quad (3)$$

with equality if and only if \mathbf{X} and \mathbf{Y} are independent. Therefore, given n random variables, $\mathbf{X}_1 \dots \mathbf{X}_n$, it holds that

$$H(\mathbf{X}_1 \dots \mathbf{X}_n) = \sum_{i=1}^n H(\mathbf{X}_i | \mathbf{X}_1 \dots \mathbf{X}_{i-1}) \leq \sum_{i=1}^n H(\mathbf{X}_i). \quad (4)$$

Given three random variables, \mathbf{X} , \mathbf{Y} , and \mathbf{Z} , the *conditional mutual information* between \mathbf{X} and \mathbf{Y} given \mathbf{Z} can be written as

$$I(\mathbf{X}; \mathbf{Y} | \mathbf{Z}) = H(\mathbf{X} | \mathbf{Z}) - H(\mathbf{X} | \mathbf{Z} \mathbf{Y}) = H(\mathbf{Y} | \mathbf{Z}) - H(\mathbf{Y} | \mathbf{Z} \mathbf{X}) = I(\mathbf{Y}; \mathbf{X} | \mathbf{Z}). \quad (5)$$

Since the conditional mutual information $I(\mathbf{X}; \mathbf{Y} | \mathbf{Z})$ is always non-negative we get

$$H(\mathbf{X} | \mathbf{Z}) \geq H(\mathbf{X} | \mathbf{Z} \mathbf{Y}). \quad (6)$$

Privacy Amplification Theorem for Noisy Main Channel

Valeri Korjik¹, Guillermo Morales-Luna², and Vladimir B. Balakirsky³

¹ Telecommunications,
CINVESTAV-IPN, Guadalajara Campus
Prol. López Mateos Sur No. 590, Guadalajara, Jalisco. Mexico
vkorjik@gdl.cinvestav.mx

² Programa de Simulación Molecular, Instituto Mexicano del Petróleo,
on leave of absence from Computer Science Section, CINVESTAV-IPN,
Av. I. P. N. 2508, 07300 Mexico City, Mexico
gmorales@cs.cinvestav.mx

³ Euler Institute for Discrete Mathematics and its Applications (EIDMA),
P.O.Box 513, 5600 MB Eindhoven, The Netherlands
v.b.balakirsky@ele.tue.nl

Abstract. Secret key agreement protocol between legal parties based on reconciliation and privacy amplification procedure has been considered in [2]. The so called *privacy amplification theorem* is used to estimate the amount of Shannon's information leaking to an illegal party (passive eavesdropper) about the final key. We consider a particular case where one of the legal parties (Alice) sends to another legal party (Bob) a random binary string \mathbf{x} through a binary symmetric channel (*BSC*) with bit error probability ε_m while an eavesdropper (Eve) receives this string through an independent *BSC* with bit error probability ε_w . We assume that $\varepsilon_m < \varepsilon_w$ and hence the *main channel* is superior to the *wire-tap channel*. To reconcile the strings between legal parties Alice sends to Bob through noiseless channel the check string \mathbf{y} based on some good error correcting code. Since this transmission is completely public Eve can eavesdrop it and therefore this extra information has to be taken into account in an estimation of the information leaking to Eve about the final key. In [3] an inequality has been proved to upper bound the information of Eve in such scenario. The main contribution of the running paper is to improve this inequality and hence to enhance the privacy amplification theorem. We present also bounds for the probability of false reconciliation when the check symbols of the linear code are transmitted through noiseless channel. The presented results can be very useful when considering the non-asymptotic case.

Keywords. Key-sharing, privacy amplification, hashing, Rènyi information, error correcting codes.

1 Introduction

We consider a cryptographic scenario where two honest persons, Alice and Bob, do not share any secret data initially but their goal is the generation of a shared *information-theoretically secure* key. Alice is able to send messages to Bob through a main *BSC* with bit error probability ε_m while *passive eavesdropper* Eve can intercept these messages

through the independent wire-tap *BSC* with bit error probability $\varepsilon_w > \varepsilon_m$. At the same time Alice is able to send messages to Bob over another binary *noiseless* but completely *public* channel. (Hence all information transmitted through this channel is known for Eve without errors.)

Such setup seems strange at a single glance but in fact it is a good model of *quantum cryptography* where a noisy *BSC* can be established due to the use of special modulation of photon flow with small intensity while noiseless channel is created by conventional amplitude modulation of photon flow with large intensity.

The *key sharing algorithm (KSA)* between Alice and Bob comprises both *reconciliation* and *privacy amplification* procedures and it is the following:

1. Alice generates truly random binary string \mathbf{x} of length k and sends it to Bob through noisy main channel.
2. Alice generates *hash function* $h(\dots)$ chosen truly randomly and sends it to Bob through noiseless public channel.
3. Alice forms the binary string $\mathbf{y} = f(\mathbf{x})$ of length $r = n - k$ as the sequence of check symbols \mathbf{y} to information symbols \mathbf{x} using some binary linear systematic error correcting (n, k) -code C with the known function $f(\dots)$, previously agreed with Bob.

We believe of course that this code C has good (the best if possible) *error correcting capability* and *constructive error correction algorithm* like *BCH* or *Goppa* codes.

4. Alice sends the check string \mathbf{y} to Bob over noiseless public channel. Eve can learn the check string \mathbf{y} received also on noiseless channel.
5. Bob receives the string \mathbf{x} as noisy version \mathbf{x}' and corrects errors on \mathbf{x}' using the check string \mathbf{y} . We believe that after such correction Bob *reconciles* \mathbf{x} and \mathbf{x}' with high probability.
6. Both Alice and Bob *hash* their strings \mathbf{x} and \mathbf{x}' (after reconciliation with \mathbf{x}) to produce the final *shared key* $\tilde{\mathbf{z}} = h(\mathbf{x})$.

The amount of *Shannon's information* I_0 leaking to Eve under the condition that she knows completely the key sharing protocol given above, the code C , the check string \mathbf{y} and the noisy version \mathbf{x}'' of the string \mathbf{x} received through wire-tap *BSC* is given by the so called *Privacy Amplification Theorem* [2]:

$$I_0 \leq \frac{2^{-(k-t_c-\ell_0)}}{\ln 2} \quad (1)$$

where t_c is the *Rényi* (or *collision*) information that has Eve from all her knowledge mentioned above, and ℓ_0 is the length of the string after hashing. The collision information t_c is comprised of the collision information t'_c about \mathbf{x} contained in \mathbf{x}'' and the collision information t''_c contained in the check string \mathbf{y} . We recall that collision information contained in \mathbf{x}'' is defined [2] as $t'_c = k - H_c(X)$, where $H_c(X)$ is the *collision entropy*. In the particular case of *BSC* as wire-tap channel with symbol error probability ε_w , the collision entropy can be calculated as

$$H_c(X) = -k \log \left(\varepsilon_w^2 + (1 - \varepsilon_w)^2 \right) \quad (2)$$

The total collision information t_c is not the sum of those particular collision informations t'_c and t''_c . To estimate an increase $\Delta_c = t_c - t'_c$ of the total collision information given side information \mathbf{y} , the following probabilistic bound can be used [3]

$$\text{Prob}(\Delta_c \leq 2r + 2s) \geq 1 - 2^{-s} \quad (3)$$

for any $s > 0$, where r is the length of the check string \mathbf{y} . In Section 2 we will improve this inequality and treat the privacy amplification theorem in more general form than in [2]. It is especially important if we are interested in a consideration of non-asymptotic case when the length k of the string \mathbf{x} is not too large.

In Section 3 we will present a modified Gallager's bound in the case of a noiseless channel for a transmission of check symbols that is just the case of our cryptographic scenario and discuss the main results.

2 Enhanced Privacy Amplification Theorem

Let \mathbf{x} be a binary uniformly distributed k -string transmitted on a communication channel from Alice to Bob during execution of step 1 in the KSA.

We propose to perform a hashing procedure $\tilde{\mathbf{z}} = h(\mathbf{x})$ presented in step 6 of KSA in two stages: Firstly, the initial string \mathbf{x} is transformed into a shorter string \mathbf{z} using a hash function chosen randomly from *universal₂ class*. It can be done as follows [7]

$$\mathbf{z} = \mathbf{x}A \quad (4)$$

where A is a truly random $k \times (\ell + r)$ -matrix, $k > \ell + r$. Secondly, the string \mathbf{z} obtained by eq. (4) is transformed to the final key $\tilde{\mathbf{z}}$ of length ℓ after the ‘‘puncturing’’ given by the transformation

$$\tilde{\mathbf{z}} = \mathbf{z}H_1 \quad (5)$$

where H_1 is a binary $(\ell + r) \times \ell$ -matrix containing exactly one 1 in each column and at most one 1 in each row. (In fact this transformation saves some ℓ digits of \mathbf{z} and deletes the remaining ones.)

Let us assume that the check string \mathbf{y} is produced at step 3 in KSA as follows

$$\mathbf{y} = \mathbf{z}H_2 \quad (6)$$

where H_2 is some binary $(\ell + r) \times r$ -matrix. All matrices A , H_1 and H_2 are public.

Theorem 1. *Under the conditions of our cryptographic scenario and the KSA presented above, there exists such a matrix H_1 that the eavesdropper's expected Shannon information I_0 , about the final key $\tilde{\mathbf{z}}$ shared by legal parties, satisfies the inequality*

$$I_0 \leq \frac{2^{-(k-t'_c-\ell-r)}}{\gamma \cdot \ln 2} \quad (7)$$

where

k is the length of the string \mathbf{x} generated by Alice in the first step of the KSA,

t'_c is the R nyi (or collision) information obtained by Eve about the string \mathbf{x} using just the string \mathbf{x}'' of her knowledge (\mathbf{x}'' is the version of \mathbf{x} received by Eve through a BSC with bit error probability ε_w),
 r is the number of check symbols sent from Alice to Bob in order to reconcile their strings and
 γ is a coefficient that approaches to 0.42 for any fixed r , as k , ℓ and $k - \ell$ increase.

In order to prove this theorem, we need to prove the following lemma:

Lemma 1. Let Z, \tilde{Z} and Y be the probability spaces that describe the random strings \mathbf{z} , $\tilde{\mathbf{z}}$ and \mathbf{y} respectively and let E be the probability space that models the eavesdropper's information on \mathbf{z} . Then, the following inequality holds

$$I(\tilde{Z}; E, Y) \leq I(Z; E) \quad (8)$$

whenever

$$\det H \neq 0, \quad (9)$$

where $H = [H_2 \ H_1]$.

Proof of the lemma. By very well known information-theoretic relations [1] we have

$$I(\tilde{Z}; E, Y) = I(\tilde{Z}; Y) + I(\tilde{Z}; E|Y) \quad (10)$$

Let us prove first that if eq. (9) were true, then $I(\tilde{Z}; Y) = 0$. Consider the joint probability $\text{Prob}[\tilde{\mathbf{z}}, \mathbf{y}] = \text{Prob}[\mathbf{z}H]$. The condition (9) implies that H is a non-singular $(\ell + r) \times (\ell + r)$ -matrix and therefore

$$\text{Prob}[\tilde{\mathbf{z}}, \mathbf{y}] = \text{Prob}[\mathbf{z} = (\tilde{\mathbf{z}}, \mathbf{y})H^{-1}] = 2^{-(\ell+r)}$$

for any $\tilde{\mathbf{z}}, \mathbf{y}$ since \mathbf{z} is uniformly distributed over $GF(2)^{\ell+r}$. On the other hand, for any \mathbf{y}

$$\text{Prob}[\mathbf{y}] = \sum_{\mathbf{z}_1 \in GF(2)^\ell} \text{Prob}[\mathbf{z} = (\mathbf{y}, \mathbf{z}_1)H^{-1}] = 2^\ell \cdot 2^{-(\ell+r)} = 2^{-r} \quad (11)$$

In a similar manner we obtain that for any $\tilde{\mathbf{z}}$

$$\text{Prob}[\tilde{\mathbf{z}}] = \sum_{\mathbf{z}_2 \in GF(2)^r} \text{Prob}[\mathbf{z} = (\mathbf{z}_2, \tilde{\mathbf{z}})H^{-1}] = 2^r \cdot 2^{-(\ell+r)} = 2^{-\ell} \quad (12)$$

Combining (11) and (12) we obtain that $\text{Prob}[\tilde{\mathbf{z}}, \mathbf{y}] = \text{Prob}[\mathbf{y}] \cdot \text{Prob}[\tilde{\mathbf{z}}]$ for any $\tilde{\mathbf{z}}, \mathbf{y}$ and hence $I(\tilde{Z}; Y) = 0$. Adding $I(E; Y) \geq 0$ on the right hand side in (10) it results in

$$I(\tilde{Z}; E, Y) \leq I(\tilde{Z}; E|Y) + I(E; Y) = I(\tilde{Z}, Y; E).$$

But $\mathbf{z} = (\tilde{\mathbf{z}}, \mathbf{y})H^{-1}$ and hence $I(\tilde{Z}, Y; E) = I(Z; E)$. This completes the lemma's proof. \square

Proof of the theorem. With condition (9), the proof of the theorem follows immediately, with factor $\gamma = 1$, from the lemma if we substitute eq. (11), where t_c means R nyi information that has Eve about \mathbf{x} from her knowledge of \mathbf{x}'' only, into the right hand

side of eq. (8) and take into account that $\ell_0 = \ell + r$. On the other hand, if we assume that $\text{rank } H_2 = r$ then there exists a set of r rows of this matrix that are linearly independent. If we choose these rows in the matrix H_1 to be zero rows and put a single 1 in the remaining rows of H_2 we obtain the relation (9). Thus the theorem is proved under the assumption $\text{rank } H_2 = r$.

By (4) and (6)

$$\mathbf{y} = \mathbf{x}AH_2 = \mathbf{x}P \quad (13)$$

where

$$P = AH_2 \quad (14)$$

Let us assign P as the matrix of the reduced echelon form representation of the generator matrix of some fixed linear binary $(k, k + r)$ code C , i. e. $[I_k P]$ is the generator matrix of C . This code (properly the matrix P) can be chosen by legal parties to be good (with large minimum code distance and having constructive algorithm of error correction).

It is easy to see that if $\text{rank } P = r$, provided $k \geq r$ (that is, P is a full-rank matrix) then it implies that $\text{rank } H_2 = r$ which is necessary to be true in order that the theorem holds. The condition $\text{rank } P = r$ may not be satisfied for any generator matrix G of the linear binary code presented in reduced echelon form as $G = [I_k P]$, but it is possible to get such condition after a number of column transpositions of G and a representation of new matrix \tilde{G} in reduced-echelon form $\tilde{G} = [I_k \tilde{P}]$ where $\text{rank } \tilde{P} = r$. (Obviously such transformations do not change the minimal code distance of the code C .)

In this manner legal parties can share initially the full rank matrix P corresponding to the “good” code and Bob will be able to correct errors in his string \mathbf{x}' after reception of check string \mathbf{y} .

But it is necessary to know the matrix H_2 in order to perform the second stage of the hashing defined by eq. (5) satisfying simultaneously condition (9). Hence a solution of matrix equation (14) is required. However, it may fail to exist. A sufficient (but not necessary) condition for its existence is the following

$$\det(A^T \cdot A) \neq 0 \quad (15)$$

where A^T is the transpose of matrix A . Under this assumption, the solution of matrix equation (14) can be obtained as

$$H_2 = (A^T \cdot A)^{-1} A^T P \quad (16)$$

The binary matrix A is a truly random $k \times (\ell + r)$ -matrix and consequently (15) is a probabilistic inequality. This probability is

$$\begin{aligned} \text{Prob} [\det(A^T \cdot A) \neq 0] &= \text{Prob} [\text{rank } A = \ell + r] \cdot \\ &\text{Prob} [\det(A^T \cdot A) \neq 0 | \text{rank } A = \ell + r] \end{aligned} \quad (17)$$

It is very easy to prove that

$$\text{Prob} [\text{rank } A = \ell + r] = \beta = \prod_{j=1}^{\ell+r} \left(1 - \frac{1}{2^{k-\ell-r+j}} \right) \quad (18)$$

On the other hand, $\alpha = \text{Prob} [\det (A^T \cdot A) \neq 0 \mid \text{rank } A = \ell + r]$ is the probability that chosen randomly matrix A^T is just the generator matrix of linear binary *complementary dual code* V (so called *LCD-code*) [4]. It means that the *hull* of the LCD-code V (defined as its intersection with its dual) has dimension 0. As it has been proven in [6], α approaches to $0.4194224417951075977099 \dots$ for any fixed r , as k and ℓ increase.

Now we have to change slightly the first stage of the hashing presented by (4). If it happens that for random chosen matrix A the condition (15) is satisfied then the next steps of the algorithm can be pursued. Otherwise, Alice should repeat a random choice of matrix A till (15) holds.

It follows from the proof of the basic privacy amplification theorem [2] that such modification of hashing procedure results in an increasing of collision probability by the factor $\frac{1}{\gamma} = \frac{1}{\alpha\beta}$ and thus it increases Shannon's information about \tilde{z} leaking Eve by the same factor. Since β is very close to 1, whenever there is a large difference between k and $\ell + r$ (a rather common situation for this algorithm) we can keep just the factor $\frac{1}{\alpha}$. This completes the proof of the theorem. \square

3 Discussion of the Main Results and Concluding Remarks

It has been proven in the previous section that for the KSA presented in section 1 and specified by formulas (4)-(6) with a slight change of the first stage of hashing when a random generation of matrix A is repeated until condition (15) is met, the upper bound (7) can be used to estimate Shannon's information leaking to eavesdropper about the final key. Practically, we can neglect by factor γ because it is close to $\frac{1}{2}$ and typically the value I_0 should be very small. Moreover we can change the KSA and believe that Alice repeats the first step of KSA using new randomly generated string x unless she meets the condition (15). Such modification results in the bound (7) with $\gamma = 1$ but the number of the main channel uses increases by $\frac{1}{\gamma}$ (a factor of 2, roughly speaking).

If the main channel is BSC with bit error probability ε_m , then asymptotically $k \cdot H(\varepsilon_m)$ check symbols sent through noiseless channel, where $H(\dots)$ is the entropy function, are sufficient to provide a reconciliation of strings x and x' with high probability. In fact, a transmission of information symbols through BSC with symbol error probability ε_m and check symbols through noiseless channel can be considered as a transmission of both groups of symbols over *time sharing channel* with the *capacity*

$$C_{ts} = \frac{k}{k+r} C_m + \frac{r}{k+r} \quad (19)$$

where $C_m = 1 - H(\varepsilon_m)$. Substitution of (19) in Shannon's condition of reliable communication [1] gives $R = \frac{k}{k+r} < C_{ts}$ which implies the condition $r \sim k \cdot H(\varepsilon_m)$. Taking into account that asymptotically [2] $t'_c \sim (1 - H(\varepsilon_m))k$ we get from eq. (7) the condition on key rate $R_k = \frac{\ell}{k}$ to provide an exponential decreasing of information I_0 leaking to eavesdropper, as $k \rightarrow +\infty$

$$R_k < H(\varepsilon_w) - H(\varepsilon_m) = C_{ks} \quad (20)$$

where C_{ks} is just the capacity of key sharing scenario under consideration [5]. When $\varepsilon_w > \varepsilon_m$ the key rate is positive and hence this algorithm works. Otherwise it is necessary

to use *public discussion* [5] to transform the condition $\varepsilon_w \leq \varepsilon_m$ into the condition $\varepsilon_w > \varepsilon_m$.

But an improvement of the bound for I_0 is important first of all in a *non-asymptotic* case. Then it is necessary to estimate the probability of errors in the string of Bob after the error correction procedure based on the use of r error free check symbols. The way of doing this is a modification of known bounds in the case when a noiseless channel is used to transmit check symbols.

Let $\lambda_k(R|P)$, where $R = \frac{k}{k+r}$ and P is a binary $r \times k$ matrix, denote the probability that the information vector \mathbf{x} of length k is incorrectly decoded by Bob based on its corrupted version \mathbf{y} and parity $\mathbf{x}P^T$ of length r . Let

$$W(\mathbf{y}|\mathbf{x}) = \varepsilon_m^{d_H(\mathbf{x},\mathbf{y})} (1 - \varepsilon_m)^{k-d_H(\mathbf{x},\mathbf{y})}$$

where d_H denotes the Hamming distance, and let

$$\mathcal{C}(\mathbf{x}, P) = \left\{ \mathbf{x}' \in \{0, 1\}^k : \mathbf{x}P^T = \mathbf{x}'P^T \right\}$$

be the coset of a linear code consisting of codewords having the same parity as the vector \mathbf{x} . Then $\lambda_k(R|P)$ can be expressed as

$$\lambda_k(R|P) = 2^{-k} \sum_{\mathbf{x} \in \{0,1\}^k} \lambda_k(R|\mathbf{x}, P)$$

where

$$\lambda_k(R|\mathbf{x}, P) = \sum_{\mathbf{y} \in \{0,1\}^k} W(\mathbf{y}|\mathbf{x}) \chi\{\beta\} \quad (21)$$

$$\beta \equiv [\exists \mathbf{x}' \in \mathcal{C}(\mathbf{x}, P), \mathbf{x}' \neq \mathbf{x} : d_H(\mathbf{x}', \mathbf{y}) \leq d_H(\mathbf{x}, \mathbf{y})]$$

and χ stands for the indicator function, i.e., $\chi\{\beta\} = 1$ if the statement β is true and $\chi\{\beta\} = 0$ otherwise.

Proposition 1. Let $\overline{\lambda_k(R|P)}$ denote the expectation of the probability $\lambda_k(R|P)$ over the ensemble of matrices P of dimension $k \times r$ whose entries are i.i.d. random variables chosen from $\{0, 1\}$ with probability $1/2$, i.e.,

$$\overline{\lambda_k(R|P)} = 2^{-kr} \sum_P \lambda_k(R|P).$$

Then

$$\overline{\lambda_k(R|P)} \leq 2^{-kE(R)} \quad (22)$$

where

$$E(R) = \min_{\rho \in (0,1]} \left[E_0(\rho) - \rho(2R - 1)/R \right] \quad (23)$$

and

$$E_0(\rho) = \rho - (1 + \rho) \log \left(\varepsilon_m^{1/(1+\rho)} + (1 - \varepsilon_m)^{1/(1+\rho)} \right)$$

is the Gallager function for a BSC with crossover probability ε_m .

Proof. Let us introduce a parameter $\rho > 0$ and let us upper-bound the probability $\lambda_k(R|\mathbf{x}, P)$ defined in (21) as

$$\lambda_k(R|\mathbf{x}, P) \leq \sum_{\mathbf{y} \in \{0,1\}^k} W(\mathbf{y}|\mathbf{x}) \left[\sum_{\substack{\mathbf{x}' \in \mathcal{C}(\mathbf{x}, P) \\ \mathbf{x}' \neq \mathbf{x}}} \left(\frac{W(\mathbf{y}|\mathbf{x}')}{W(\mathbf{y}|\mathbf{x})} \right)^{1/(1+\rho)} \right]^\rho \quad (24)$$

If $\rho \in (0, 1]$, then

$$\begin{aligned} & 2^{-kr} \sum_P \left[\sum_{\substack{\mathbf{x}' \in \mathcal{C}(\mathbf{x}, P) \\ \mathbf{x}' \neq \mathbf{x}}} W^{1/(1+\rho)}(\mathbf{y}|\mathbf{x}') \right]^\rho \\ & \leq \left[2^{-kr} \sum_P \sum_{\substack{\mathbf{x}' \in \mathcal{C}(\mathbf{x}, P) \\ \mathbf{x}' \neq \mathbf{x}}} W^{1/(1+\rho)}(\mathbf{y}|\mathbf{x}') \right]^\rho \end{aligned} \quad (25)$$

$$\begin{aligned} & = \left[\sum_{\mathbf{x}' \neq \mathbf{x}} W^{1/(1+\rho)}(\mathbf{y}|\mathbf{x}') 2^{-kr} \sum_P \chi\{\mathbf{x}' \in \mathcal{C}(\mathbf{x}, P)\} \right]^\rho \\ & = \left[\sum_{\mathbf{x}' \neq \mathbf{x}} W^{1/(1+\rho)}(\mathbf{y}|\mathbf{x}') 2^{-r} \right]^\rho \end{aligned} \quad (26)$$

$$\leq 2^{-\rho r} \left(\varepsilon_m^{1/(1+\rho)} + (1 - \varepsilon_m)^{1/(1+\rho)} \right)^{k\rho}. \quad (27)$$

Inequality (25) follows from Jensen's inequality and (26) holds because of the linearity:

$$2^{-kr} \sum_P \chi\{\mathbf{x}' \in \mathcal{C}(\mathbf{x}, P)\} = 2^{-kr} \sum_P \chi\{(\mathbf{x} \oplus \mathbf{x}')P^T = (0, \dots, 0)\} = 2^{-r}$$

where $(0, \dots, 0)$ is the vector consisting of r zeroes.

Thus, (24) and (27) imply

$$2^{-kr} \sum_P \lambda_k(R|\mathbf{x}, P) \leq \min_{\rho \in (0,1]} 2^{-\rho r} \left(\varepsilon_m^{1/(1+\rho)} + (1 - \varepsilon_m)^{1/(1+\rho)} \right)^{k(1+\rho)}$$

and (22) follows.

The function $E(R)$ is given in Fig. 10 for a BSC with crossover probability 0.01. Note that this function is always positive if

$$R < \frac{1}{1 + \varepsilon_m \log \varepsilon_m + (1 - \varepsilon_m) \log(1 - \varepsilon_m)}$$

Note also that if $\rho = 1$, then (22) becomes the union bound :

$$\overline{\lambda_k(R|P)} \leq \left(1 + 2\sqrt{\varepsilon_m(1 - \varepsilon_m)} \right)^k 2^{k(1-R)}.$$

These approaches can be also used to estimate the probability $\lambda_k(R|P)$ for specific matrices P .

Thus, we have some tools of analysis that allow us to assign the parameters k, r , and ℓ to provide both reliable and secure key sharing between legal parties in the presence of passive eavesdropper.

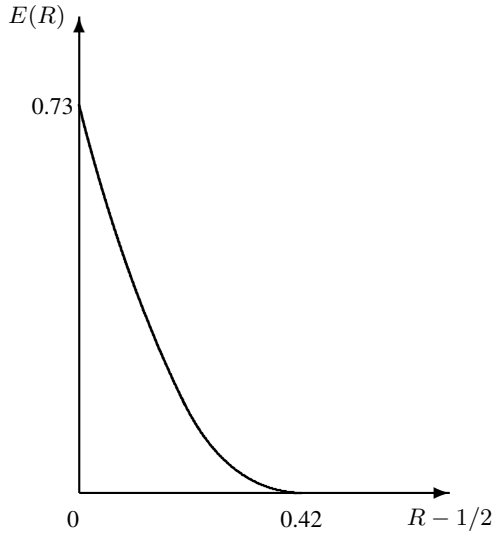


Fig. 1. Exponent $E(R)$ defined in (23) for a BSC with crossover probability 0.01.

References

1. Ash, R.B. “*Information Theory*”. Dover, New York, 1990.
2. Bennett, C. H., Brassard, G., Maurer, U. M. “Generalized Privacy Amplification”. *IEEE Trans. on IT*, vol. 41, nr. 6, pp. 1915-1923. 1995.
3. Cachin, C., Maurer, U. M. “Linking Information-Reconciliation and Privacy Amplification”. *Eurocrypt'94: Advances in cryptology, Lecture Notes in Computer Science*, vol. 950, pp. 267-274. Springer-Verlag. 1995.
4. Massey, J.L. “Linear Codes with Complimentary Duals”. *Discrete Math.*, 106/107. 1992.
5. Maurer, G. “Secret Key Agreement by Public Discussion Based on Common Information”. *IEEE Trans. on I. T.*, vol. 39, nr. 3, 1998, p. 733-742.
6. Sendrier, N. “On the dimension of the hull”. *SIAM Journal on Discrete Mathematics*, vol. 4, nr. 2, pp. 282-293, 1997.
7. Stinson, D.R.. “Universal Hashing and Authentication Codes”. *Advances in cryptology, Crypto'91, Lecture Notes in Computer Science*, vol. 576, 1992, pp. 74-85. Springer-Verlag. 1992.

Efficient Kerberized Multicast in a Practical Distributed Setting

Giovanni Di Crescenzo¹ and Olga Kornievskaja²

¹ Telcordia Technologies, Inc., NJ, USA*
giovanni@research.telcordia.com

² CITI, University of Michigan, MI, USA
aglo@eecs.umich.edu

Abstract. Many of today's distributed applications are based on group communication. Given the openness of today's networks, communication among group members must be secure and, at the same time, efficient. In this paper we consider a realistic distributed setting modeling general networks, including the Internet, that suggests the use of Kerberos, and, more specifically, a distributed mode of operation of Kerberos, called *crossrealm authentication protocol*, as a desirable choice for distributed applications.

We design an efficient protocol for secure group communication via multicast, using Kerberos. While developing the main components of our protocol, we construct an efficient paradigm for crossrealm authentication protocols that decreases communication over the Internet, makes most operations local, and reduces the workload of the involved components. We also design extensions of single-center multicast encryption schemes to multiple-center schemes. Our main protocol is obtained by combining these two contributions.

1 Introduction

Audio and video conferencing, data casting, and collaborative applications are among the next-generation applications emerging on the Internet that are based on group communication. When communication is conducted over a wide-area network such as the global Internet, security is crucial as messages traverse many links that are prone to attacks. Group members must have the means to communicate securely, in a private and authentic manner. While peer-to-peer security is a well-developed field, and several practical security protocols and primitives have been proposed, the problem of designing practical and secure group communication protocols is not as well explored.

In this paper we study, in a practical distributed setting motivated by real-life and developed applications, the problem of secure group communication. We now describe the setting we consider, the tools used and previous results, and finally describe our results.

* Copyright 2001, Telcordia Technologies, Inc. All Rights Reserved.

Model considered. In this paper we consider security protocols for a practical distributed setting. Specifically, we consider several users that are divided into realms or Intranets, all of these being connected through the Internet. Moreover, in each Intranet there is a server or center responsible for several operations related to security management within its own Intranet. We assume that the communication through the Internet is not secured by default. This setting is sufficiently general, as almost any type of network can be characterized as described, and, yet, offers a useful structure in the form of a trusted server inside every Intranet.

Tools used and previous results. Two basic tools we use are: Kerberos protocol and multicast encryption schemes.

Kerberos, a widely used authentication mechanism, is a key distribution protocol that states how to share a secret between two participants. Crossrealm authentication is an extension to the original protocol that states how to deal with authentication in a distributed environment. One interesting feature of Kerberos is that almost all of its design is based on symmetric-key cryptography and does not suffer from computational overhead or open status of certain public-key cryptography techniques (as, for instance, digital signature revocation). Kerberos relies on the existence of a trusted server, which may limit its application to settings where this assumption is realistic. On the other hand, the setting we consider already satisfies this assumption. This, among other factors, has strongly motivated our choice of the Kerberos system. Although Kerberos (originally designed as a third party protocol) has extensions to distributed settings, we found out that the efficiency of some of these extensions can be questioned, thus leaving room for possible improvement.

Our results. In devising a practical and secure solution to our problem, we rule out the use of public-key crypto because of its enormous cost relative to symmetric-key crypto. Public-key techniques are suitable in a setting with no trusted centers. However, as our setting includes trusted centers, a combination of efficient extensions of Kerberos to a distributed environment and efficient extensions of multicast encryption schemes offers more practical solution of our problem.

Globally speaking, the contribution of our paper is the construction of a practical protocol for secure group communication via multicast using Kerberos.

In developing this solution, we have designed extensions to Kerberos and used some recently obtained extensions to multicast encryption schemes. In particular, we have designed a new practical paradigm for crossrealm Kerberos authentication that significantly decreases the amount of slow communication over the Internet, makes most operations local, and reduces the workload of the involved components. The scheme extends efficiently to handle multicast encryption. Moreover, we use some new extensions of multicast encryption schemes to the case of multiple centers, which is relevant in our setting.

Our main protocol is then obtained as a careful combination of the new paradigm for crossrealm Kerberos authentication with the schemes for multi-server multi-

cast encryption. While performing this combination, we pay special attention to communication and storage complexity.

Organization of the paper. The rest of the paper is organized as follows. In Section 2, we present an overview of the Kerberos authentication protocol. In Section 3, we present an overview of crossrealm authentication in Kerberos. In Section 4, we describe multicast encryption schemes, where we recall the notion and basic solution approach and then show, in Sections 4.1 and 4.2, the two main paradigms of key management schemes for multicast encryption. Readers familiar with the background material should skip to Section 5, where we introduce an efficient crossrealm authentication protocol. In Section 6, we design extensions of known multicast encryption key management protocols to the environment of many multicast servers. In Section 7, we show how to obtain our protocol for secure group communication by carefully integrating the previous schemes, namely, the new crossrealm Kerberos authentication protocols and the new multi-server multicast encryption schemes.

2 Overview of Kerberos

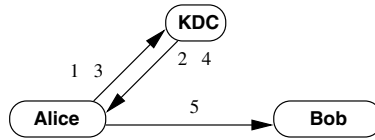
Kerberos [9] is a network authentication system based on the Needham-Schroeder protocol [8]. To avoid quadratic explosion of prior key agreement, Kerberos relies on a trusted third party, referred to as a Key Distribution Center (KDC). *Alice*, a Kerberos principal, and *Bob*, a Kerberized service, each establish shared secrets with the KDC.

Authentication is achieved when one party proves knowledge of a shared secret to another. Kerberos authentication process makes use of two special data structures: a ticket and an authenticator. A ticket is created by a KDC. It contains a secret key intended for two entities that will engage in authentication. The ticket and the secret key are given to a requestor. To authenticate the requestor creates an authenticator by using the secret key received from the KDC. Kerberos authentication is illustrated in Figure 1.

Let us consider an example, where *Alice* wishes to authenticate to *Bob*. At login, *Alice* receives a ticket granting ticket, TGT, from the KDC. She uses her password to retrieve a session key encrypted in the reply. The TGT allows *Alice* to obtain tickets from a Ticket Granting Service, TGS, for other Kerberized services. To access a Kerberized service, *Alice* authenticates to the TGS. She presents her TGT and constructs the authenticator, $\{T\}_{K_{A,TGS}}$. If authentication is successful, *Alice* receives a service ticket, $\{Alice, Bob, K_{A,B}\}_{K_B}$. To authenticate to *Bob*, *Alice* constructs an authenticator, $\{T\}_{K_{A,B}}$, proving to *Bob* that she knows the session key inside of the service ticket.

3 Overview of Crossrealm Authentication in Kerberos

In this variant of Kerberos, there are several KDCs, one for each realm. In addition to assuming that there is a shared key between each user and their local

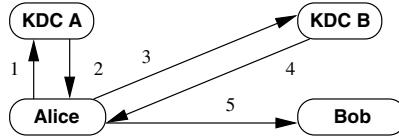


LOGIN PHASE:	ONCE PER SESSION
1. <i>Alice</i> → <i>KDC</i> :	"Hi, I'm <i>Alice</i> "
2. <i>KDC</i> → <i>Alice</i> :	$TGT = \{Alice, TGS, K_{A,TGS}\}_{K_{TGS}}, \{K_{A,TGS}, T\}_{K_A}$
ACCESSING SERVICES: EVERY TIME BEFORE TALKING TO A SERVICE	
3. <i>Alice</i> → <i>TGS</i> :	<i>Alice, Bob, TGT, $\{T\}_{K_{A,TGS}}$</i>
4. <i>TGS</i> → <i>Alice</i> :	$TKT = \{Alice, Bob, K_{A,B}\}_{K_B}, \{K_{A,B}, T\}_{K_{A,TGS}}$
5. <i>Alice</i> → <i>Bob</i> :	"Hi, I'm <i>Alice</i> ", <i>TKT, $\{T\}_{K_{A,B}}$</i>

Fig. 1. Kerberos authentication. Two phases are shown: initial authentication and service ticket acquisition. KDC is the Kerberos Key Distribution Center. TGS is the Ticket Granting Service. Most implementations combine these services. K_{TGS} is a key shared between the TGS and the KDC. K_A is a key shared between *Alice* and the KDC, derived from *Alice*'s password. $K_{A,TGS}$ is a session key for *Alice* and TGS. $K_{A,B}$ is a session key for *Alice* and *Bob*. T is a timestamp used to prevent replay attacks. In each step, the encrypted timestamp serves the role of an authenticator. Note the version of Kerberos in this figure does not include an optional pre-authentication in the initial authentication. Furthermore, the steps shown describe only one way authentication, *Alice* to *Bob*. Mutual authentication is achieved with an additional message from *Bob* to *Alice*.

KDC, this protocol assumes that there is a shared key between any two KDCs that participate in the crossrealm authentication. Even if the original Kerberos protocol doesn't specify how secret keys are established between participating KDCs, any of the solutions in the public-key cryptography literature for sharing keys among two parties can be used. An example of an use of such a solution can be found in an extension of Kerberos, called PKCROSS [5].

Once again, suppose *Alice* wishes to communicate with *Bob*. Crossrealm authentication proceeds as illustrated in Figure 2. As before, there is an initial login phase where *Alice* authenticates to her local KDC (this phase is not shown in the picture as it is identical to the original Kerberos protocol). *Alice* uses her TGT to request a remote ticket granting ticket (RTGT) for the remote realm. A local KDC Kerberos authenticates the request by verifying the validity of the ticket and the authenticator. The KDC constructs the RTGT and encrypts it with a key shared between the two KDCs; in our example, it's KDC A and KDC B. *Alice* presents the RTGT to KDC B and requests a service ticket for *Bob*. KDC B checks that RTGT is valid and that *Alice* included the corresponding authenticator, $\{T\}_{K_{A,KDCB}}$. It proceeds by issuing a service ticket.



1-2	Kerberos login
-----	----------------

3. *Alice* → *KDC A*: *Alice*, *Bob@B*, TGT, $\{T\}_{K_{KDCA}}$
 4. *KDC A* → *Alice*: RTGT = $\{Alice@A, K_{A,KDCB}\}_{K_{KDCA,KDCB}}, \{K_{A,KDCB}, T\}_{K_{A,KDCA}}$
 5. *Alice* → *KDC B*: *Alice@A*, *Bob@B*, RTGT, $\{T\}_{K_{A,KDCB}}$
 6. *KDC B* → *Alice*: TKT = $\{Alice@A, Bob@B, K_{A,B}\}_{K_B}, \{K_{A,B}, T\}_{K_{A,KDCB}}$
 7. *Alice* → *Bob*: “Hi, I’m *Alice@A*”, TKT, $\{T\}_{K_{A,B}}$
-

Fig. 2. Crossrealm authentication. Initial Kerberos authentication (steps 1-2) is not shown in the figure. Also, a secret key, $K_{KDCA,KDCB}$, shared between the KDCs is assumed to be established prior to clients’ requests. *Alice* authenticates with her local realm by presenting the TGT and constructing the authenticator for which she uses $K_{A,KDCA}$, key shared between *Alice* and the KDC A. *Alice* receives a remote ticket granting ticket, RTGT, encrypted with $K_{KDCA,KDCB}$. *Alice* proceeds to request a service ticket from the remote KDC. The last two steps of the protocol are equivalent to the last two steps of the original Kerberos.

4 Multicast Encryption with a Single Server

The word *multicast* is often associated with a variety of communication and networking topics; in this paper we consider *multicast encryption*. Informally, multicast encryption schemes are methods for performing secure communication among a group of users. In such schemes the group structure is dynamically changing, as a consequence of additions and deletions of members. Moreover, they are divided into two phases: an off-line initialization phase, in which a center communicates privately with each group member, and an on-line communication phase, in which the only communication means between center and group members is through a multicast channel, which the center (or any other party) uses to broadcast messages to all others.

Given that the communication is secured through symmetric-key encryption we consider the usual paradigm of sharing a random key among the group members. Once established, this *group key* is used to encrypt and decrypt messages exchanged among members of the group. The problem is then reduced from a multicast encryption to a *key management problem*. The latter requires a scheme for managing keys among the group members in such a way that the following invariant is maintained: all group members have knowledge of the group key and all others have no information about it, where the group membership is not fixed, but varies through operations such as addition and deletions of users.

Key management protocols in the literature can be divided into two types: collaborative and non-collaborative. *Collaborative* key management refers to a method of managing the group key between the group members without any outside assistance; collaborative key management protocols have been proposed, for instance, in [10][16]. *Non-collaborative* key management, which we consider in this paper, refers to a method that relies on the existence of a *center* that is responsible for generating, distributing and updating the group key and other keys in the system.

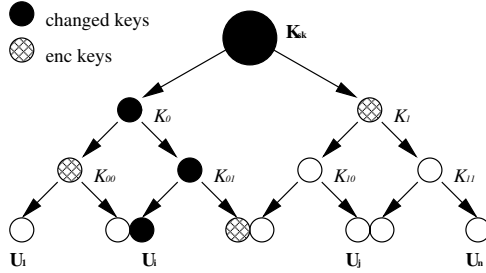
In the literature, all non-collaborative key management architectures consider the scenario of a single center managing N users that comprise a multicast group. This center, which we call C , is responsible for managing keys and distributing them to users U_i , for $i = 1, \dots, N$, both in an initial phase and while they are joining and leaving the group.

Joining a group is inexpensive if no backward secrecy is required. Typically, when a user wants to join a group, he contacts a center and receives a group key encrypted with a secret key established previously (like a password in Kerberos). If backward secrecy is required, then a new group key is created and distributed to all group members including the newly joined user. When a user wants to leave a group, the group key must be changed in order to satisfy the stated invariant. A center then generates a new group key and distributes it to the group. Multicast encryption schemes differ in how the key update operation is managed. In this paper we assume that backward secrecy is not needed and thus the join operation is simple. We mostly concentrate on describing what happens when a user leaves a group.

The multicast encryption schemes presented in the literature are compared based on several complexity measures, such as communication complexity, storage complexity both for the user and the center, and time complexity. Although all such complexity measures are important, given the type of applications multicast encryption schemes are usually employed for, it is often of crucial importance to design a scheme having small communication complexity (for instance, at most polylogarithmic in the number of users, for each update operation). For more extensive discussions, we refer the reader to [23]. In the rest of this section we briefly review the two main families of multicast encryption schemes in the literature: minimal storage schemes and tree-based schemes. Their performance, in the various complexity measures, is summarized in the table in Figure 3.

4.1 Minimal Storage Schemes

A typical scheme in this family works as follows. Each user U_i holds two keys: a group key K_g and a unique secret key K_i shared between this user and the center. The center holds two keys: a group key K_g and a secret key l . This secret key is used to generate the keys for all the users in the group by combining it with a pseudo-random function f , for instance, as $K_i = f_l(i)$. We now describe how the center manages keys when users join or leave the group, the latter case being the non-trivial one.



Evaluation parameters	Minimal storage	Basic tree	Improved tree
user storage	2	$\log n + 1$	$\log n$
center storage	2	$2n - 1$	$\frac{n}{\log n}$
communication	$n - 1$	$\log n$	$\log n$
computation	n	$\log n$	$\log n$

Fig. 3. Basic tree scheme. The figure shows the tree created by the center in the basic tree scheme, for $N = 8$. A highlighted path represents the affected path after a user deletion. Keys of the affected nodes are shaded. The table above compares three multicast encryption schemes.

When a user, say U_{N+1} , joins the group, the center just computes a key $K_{N+1} = f_l(N + 1)$ and gives it to U_{N+1} ; then the center privately sends the group key to U_{N+1} by first encrypting it using key K_{N+1} .

When a user leaves the group, the center chooses a new group key $K_{g'}$, and, for each user U_i who has not left the group, the center encrypts $K_{g'}$ using K_i as a key and transmits the result to U_i . The communication cost of this scheme is linear in the number of clients in the group. The storage cost of both user and center is constant.

4.2 Basic Tree Schemes

These schemes reduce the communication cost of key update to logarithmic but increase the storage cost. The center creates a balanced tree. For simplicity, we assume that for n members, $n = 2^t$ for some natural number t . A key is associated with each node in the tree. Leaf nodes are associated with individual keys known to a particular user. Each user is given all the keys associated with nodes on the path from the leaf node to the root. Thus, each user U_i holds $\log n + 1$ keys. The root node is considered to be a group key for the group since it is known to all the users. The center stores all the keys in the tree, requiring a storage cost of $2n + 1$ keys. We now describe how the center manages keys when users join or leave the group; again, the latter case is the non-trivial one.

When a user joins the group, the center adds a new leaf to the tree, associates a new key and the new member with the new leaf, and distributes all keys on the path from this leaf to the root to the new user.

When a user leaves the group, the center changes all the keys on the path from the removed leaf to the root. The center generates $\log n$ new keys. Each of the keys is encrypted under the key of the unaffected sibling node on the path and multicasted to all users. Overall, only $\log n$ encrypted keys are sent.

Figure 3 shows an example of key assignments in the basic tree scheme for $N = 8$. In addition to 8 leaf nodes, there are 7 other keys in the system. User U_3 stores 4 keys: K_{010} , K_{01} , K_0 , and K_{sk} . If user U_3 leaves, then keys K_{01} , K_0 , and K_{sk} must change. The new key K'_{01} is encrypted with K_{011} . The new key K'_0 is encrypted with K_{00} and K'_{01} . The new session key is generated next and encrypted with K'_0 and K_1 . This technique guarantees that new keys will be known only to the rest of the group and not to the removed user.

This scheme takes advantage of the fact that some keys are shared between multiple clients and are not on the affected path. The details of the update protocol are described in [21, 23]. An improved binary tree scheme is described in [3], which generalizes from binary trees to a -ry trees and applies a minimal storage scheme on the groups of users of size m at the bottom of the tree. In the version of this scheme that achieves efficient communication $O(\log n)$ and user storage $O(\log n)$, the center storage is improved from linear to $O(n/\log n)$.

5 Efficient Crossrealm Authentication Protocol

In Section 3, we showed how users from different Kerberos realms can authenticate each other. We note that the crossrealm authentication protocol that was discussed requires the user to contact the remote KDC before being able to contact the other user. As a consequence, in the (typical) situation in which each user is in her own Intranet and the two Intranets are connected through Internet links, the crossrealm authentication protocol introduces network delays associated with the traffic going through Internet links, which are potentially much slower than Intranet links. The *fake ticket protocol* described in this section reduces the network delay seen by the client.

Our protocol maintains the same properties of the original crossrealm authentication protocol. Specifically, our protocol provides a mechanism for two parties to establish a secret key and authenticate one another in the process. For now we make an assumption that the participating KDCs share a secret key. The same assumption is made in the original crossrealm authentication protocol. Additionally, we assume that clients trust KDCs to make requests on their behalf. The KDC is a trusted entity (recall that this is a basic assumption of Kerberos itself), so this assumption is reasonable.

Figure 4 gives details of the protocol, including the flow of messages among the participants. First, we give intuition behind the protocol; then we go over each of the steps in detail. As in the original crossrealm protocol, *Alice* makes a request

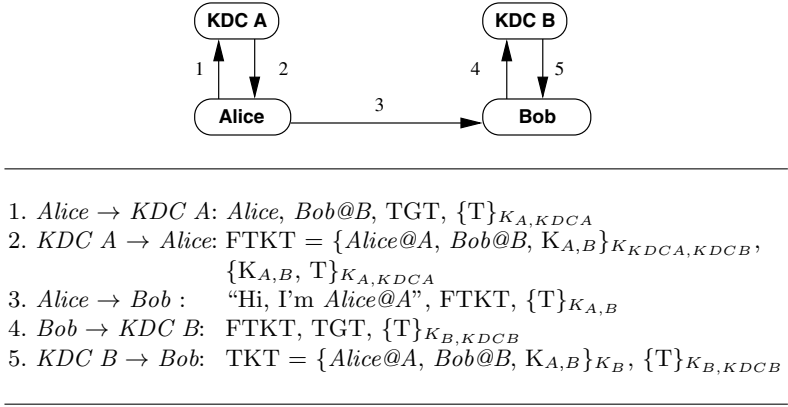


Fig. 4. Fake ticket protocol. The figure shows the flow of messages between participants in the protocol. *Alice* authenticates to the KDC A by presenting the TGT and creating the corresponding authenticator. KDC A generates a fake service ticket, (FTKT), encrypts the session key with the key it shares with *Alice*. She authenticates to *Bob* with the ticket, (TKT), and the authenticator. *Bob* forwards the ticket to the KDC B which after authenticating *Bob* by validating his TGT and the included authenticator creates a real service ticket.

to her local KDC for a service ticket. The local KDC replies to the user with a service ticket for *Bob*. *Alice* continues and makes a request to *Bob*, as in the original Kerberos protocol.

Notice that in this protocol *Alice*'s local KDC can not issue a valid service ticket for *Bob*, as he is not in the same realm as *Alice*, therefore, *Alice*'s KDC does not know the shared key between *Bob* and *Bob*'s local KDC. Instead, *Alice*'s local KDC issues a *fake* ticket. When a KDC issues any kind of a ticket to an entity, the requestor has no way of verifying the validity of the ticket because the ticket is encrypted with a key that is not known to that entity. *Alice* can not know that the ticket she has is not encrypted with the proper key. *Alice*'s KDC generates a session key to be used by *Alice* and *Bob* and gives this key to *Alice*.

Upon receiving a service ticket from *Alice*, *Bob* is unable to decrypt it because he doesn't share any secrets with *Alice* or *Alice*'s KDC. So *Bob* contacts his local KDC and requests verification of the ticket. The KDC extracts the session key from the ticket and returns it to *Bob*.

In step 1, *Alice* makes a request for a service ticket from her local KDC. The format of the message is as in previous protocols. The local KDC authenticates the client and generates a session key to be used by *Alice* and *Bob*. It encrypts the session key with the key it shares with *Alice*. Also it generates a ticket and sends the reply to *Alice*. The ticket is encrypted with a key that is shared between *Alice*'s KDC and *Bob*'s KDC, $K_{KDCA,KDCB}$.

In step 3, upon receiving the reply from her KDC, *Alice* creates a request to *Bob* that is identical to the last step of the Kerberos protocol (see Figure 1). The request contains the service ticket for *Bob* that *Alice* has just received and the authenticator, $\{T\}_{K_{A,B}}$, proving to *Bob* that she knows the session key inside of the service ticket. The request is marked as crossrealm authentication request.

In step 4, when the server gets the message that contains this special type of a ticket, *Bob* forwards the ticket to his local KDC B. *Bob* authenticates to the KDC B by presenting his ticket and the authenticator that proves that *Bob* knows the session key that goes with the ticket.

In step 5, after successful authentication of *Bob*, KDC B decrypts the ticket and retrieves the session key. KDC B also checks that the service ticket was intended for *Bob* before revealing the session key inside of the ticket. Furthermore, it checks that the ticket is still valid by checking the expiration time. KDC B creates a real service ticket for *Alice* and *Bob* and returns this ticket to *Bob*.

Security analysis. The proposed protocol is a variation of the original crossrealm protocol. Thus, if the original protocol is secure then so is the *fake ticket* protocol. At each step of the protocol the sender (and optionally the receiver) is authenticated with the corresponding tickets and authenticators. Furthermore, in order to prevent *Charlie* from capturing a *fake* ticket and asking the KDC to reveal the session key inside of it, we require that the party authenticated in step 4 of the protocol be the same principal for which the *fake* ticket was issued.

5.1 Comparison of Protocols

In the previous sections, we presented two crossrealm authentication protocols. One was previously known and one was introduced in this paper. Both protocols achieve the goal of establishing a secret key between communicating parties and authenticating each other in a setting where parties are divided into realms. Each realm has a server responsible for various operations within a realm, such as key management.

This section looks at how the protocols introduced in this paper compare to the protocol previously introduced in the literature. We consider the following comparison criteria: communication delays, software compatibility, workload at the client side, workload at the KDCs (local and remote) for each of the protocols, and efficient multicast applicability.

Communication delays. To compare communication cost, we identify two types of messages: Intranet (fast) and Internet (slow). Overall, we observe that the *fake ticket* protocol is the most favorable. Each execution of the original crossrealm protocol requires three Internet messages, while the *fake ticket* protocol requires only one. Intranet messages are assumed to be fast, so the difference in their numbers (in our case two messages) should not affect overall performance.

Software compatibility. Each of the crossrealm protocols requires certain modifications to Kerberos. To support crossrealm operations in Kerberos, client software needs to be modified to handle communication with the remote KDC. The *fake ticket* protocol doesn't require client side modifications. In practice, it is very important that the client side software remains unchanged.

Each of the protocols requires modifications to the KDC. In the original crossrealm proposal, the remote KDC must recognize remote ticket granting tickets. In practice, this is not a major modification. In the *fake ticket* protocol the ability to generate fake tickets is required. Furthermore, the support for converting fake tickets into regular tickets is needed. In practice, the support for additional functionality is implemented in terms of new services that run on the same machine as the KDC and thus do not require modification to already existing code.

In addition to these modifications, the *fake ticket* protocol requires modifications to the server software.

Client workload. We measure the workload of a component in terms of the number of cryptographic operations and network messages. The original crossrealm protocol produces the biggest client workload. If the client machine is not powerful enough to do encryption operations, then overall performance of the system degrades. In the original crossrealm protocol the client is required to perform seven cryptographic operations (encryption/decryption) and process five messages. The *fake ticket* protocol requires four cryptographic operations and three messages.

KDC workload. In case of the remote KDC, all protocols produce the same number of encryptions, which is not surprising as the remote KDC is the only one that can generate a valid service ticket for the server. This operation takes three encryptions. The request for a service ticket is authenticated by a ticket and a nonce. In order to decrypt a ticket and a nonce, a KDC performs two decryption operations.

Efficient multicast applicability. As it will become clear in Section 7, the *fake ticket* protocol can be extended to allow multicast encryption by preserving its efficiency in all parameters. The original crossrealm protocol can not be easily combined with multicast encryption to produce a protocol that is reasonably efficient.

6 Multicast Encryption with Many Servers

As we pointed out in Section 3, the solution that assumes a single server responsible for all the clients doesn't scale to the environments such as an Internet connecting several Intranets (as well as the Internet itself). The previously reviewed schemes for multicast encryption consider only a single center. In this

section, we discuss several protocols for the more realistic case where there are multiple centers, each managing a group of users. In Table 1 we summarize performance characteristics of each of the schemes.

Imagine a dynamically-changing group of n users (namely, with users being added to and removed from the group); as before each client is denoted as U_i . A group is broken down into smaller groups of size m . Each such group is managed by a center, denoted as C_j .

The security requirements that a *multi-server secure multicast encryption scheme* should satisfy can be defined essentially in the same way as those of a single-server scheme. The only difference is that in the various operations of a multi-server protocol, the users may potentially interact with many servers (rather with only one), and the servers may have to coordinate in order to perform all necessary operations. As a result, users managed by different centers can belong to the same group.

A straight forward approach in constructing a multi-server multicast encryption scheme, given a single-server one, elects one out of the many servers as a principal server and then uses it as the server when executing the single-server scheme. The inconvenience of such a scheme is that for every operation, each client must communicate to a server in a specific Intranet, which, in general, requires Internet messages to be sent, degrading overall performance.

Instead, we would like multi-server schemes to satisfy some basic efficiency requirements. First, we require every user to communicate only to the *local* server in the same Intranet during the various management operations of the scheme. Moreover, we require both the total communication among the servers and the storage complexity of all servers to be minimized. This list of requirements is not exhaustive: it is not hard to think of other efficiency requirements that might be necessary for certain specific applications.

Table 1. Multi-server multicast encryption schemes. The table summarizes the complexity values for each of the multi-server multicast encryption schemes. Values for center storage are of a single center. To get the total center storage among all the centers each of the values need to be multiplied by number of centers, assumed for the first two case to be $\frac{n}{m}$.

Parameters	Replicated tree	Master key	Weighted tree
User storage	$\log n + 1$	$\log m + 1$	$O(\log n)$
Center storage	$2n - 1$	$2m$	$O(m + \log \frac{n}{m})$
Communication	$\frac{n}{m} \log n$	$\log m + \frac{n}{m}$	$O(\log n)$

6.1 A Simple Construction

A simple construction groups all users into a single binary tree, as in the single server tree based schemes, and then requires that the same tree data structure is maintained by each of the centers. Addition of a user requires the user to communicate only with its local center, who later communicates the join to the other centers. Similarly, deletion of a user is handled locally by the center associated with the departing user, as for a single center scheme, and later the center informs all other centers of the deletion.

Although obtaining locality of key updates (the communication for each update is efficient, as it is logarithmic in n), this solution is inefficient in terms of center storage. Each center is required to store all the client keys, including the keys of the clients it doesn't manage. The total number of keys stored among all the centers for a group is $(2n - 1) \cdot \frac{n}{m}$, assuming there are $\frac{n}{m}$ centers, each managing m out of n clients.

6.2 A Second Construction, Extending Tree-Based Schemes

To improve on the previous scheme, we observe that each of the centers doesn't need to keep the information about the users that don't belong to the same Intranet as that center. So if each center manages m users, then each of the centers applies the single center binary tree scheme to store information about their users only. As a consequence, each user stores only $\log m + 2$ keys; a center stores the keys in the tree for m clients. The root of each tree is a key shared by m users and not all n users. Thus each client additionally stores a session key for the group, K_{sk} . Each of the centers stores a secret key, K_{mk} , not known to the clients, that is used to distribute a new session key.

When a user leaves, his local center creates a new session key for the whole group. Distribution of the session key to the local users is done by applying the single center tree-based protocol. The shared key among the centers is not affected, so the center uses it to encrypt the new session key and broadcasts it to the rest of the centers. Upon receiving the message and decrypting it, each of the centers encrypts with the corresponding root key for the local tree and broadcasts it. The update operation takes $\log m + \frac{n}{m}$ messages. The total storage among all centers is $2(m + 1) \frac{n}{m}$.

We note that this scheme improves over the previously described scheme both in center and in user storage, but has a more expensive communication complexity for each key update.

6.3 A Third Construction, Based on Coding Theory Algorithms

One possible extension of the previous scheme is to create a better data structure for the keys shared among the servers. Specifically, a binary tree having centers as leaf nodes could help. Note that the resulting data structure of the entire system contains several binary trees, each in a different Intranet, built over the

users in that Intranet and having a center as a root; and, moreover, one binary tree over each of the centers. Overall, the data structure is in fact a single binary tree; however, unlike the case in which a binary tree is obtained by applying a single center tree-based scheme, here the tree can be unbalanced if the relative numbers of users in each Intranet differ significantly from each other.

Consequently, the problem of finding the optimal balancing, given as information the number of users in each Intranet, naturally rises as a way to minimize the average number of keys to be changed in the next update operation (assuming that the next user leaving is random). Interestingly, this problem can be seen to be equivalent to the well-known coding theory problem of finding codes for variable-length sources that minimize the average codeword length. A classical algorithm constructing such codes, based on the is due to Huffman (see, e.g., [7]). This algorithm, given their frequencies, constructs a tree which minimizes the average length of the codewords.

To construct an optimal multicast tree for our scheme, we first construct a tree in each Intranet using a single center tree-based scheme. Then, each center is associated with a codeword. The number of users in that Intranet divided by the number of total users is considered as the frequency of that codeword. We run Huffman's algorithm on these inputs and obtain a minimal tree whose structure shared among all clients and all servers and added to all trees previously constructed for each Intranet. This gives rise to a single, global tree.

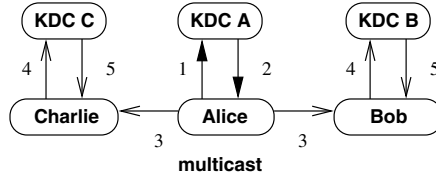
Addition and deletion of users is handled as in the single center tree-based scheme, but on the global tree.

We note that in this scheme each user is given at most $\log n$ keys; some come from the tree constructed for that Intranet, and some from the execution of Huffman's algorithm, the relative sizes according to how many users are in the same Intranet. The communication per update is now only logarithmic in n , and is, in fact, optimal assuming that the next deletion is from a randomly chosen user. The total center storage is also efficient, although the exact expression depends on the relative numbers of users in each Intranet; in the case when all Intranets have the same number m of users, each server stores only $\log(\frac{n}{m}) + m$ keys.

7 Integrating Kerberos with Multicast Encryption Schemes

In this section we combine two parts of this paper into a single protocol for secure group communication via multicast using Kerberos-based protocols. This protocol is designed into the practical distributed setting considered in this paper; namely, users are divided into realms or Intranets, which are connected through the Internet.

We show how to combine both of the crossrealm protocols with any of the discussed multi-server multicast encryption schemes. For simplicity of description,



-
1. $Alice \rightarrow KDC\ A:$ $Alice, Group, TGT, \{T\}_{K_{A,KDCA}}$
 2. $KDC\ A \rightarrow Alice:$ $GTKT = \{Alice@A, Group, K_{gk}\}_{K_{mk}}, \{K_{gk}, K_{help}^A, T\}_{K_{A,KDCA}}$
 3. $Alice \rightarrow Group:$ "Hi, I'm Alice", $GTKT, \{T\}_{K_{gk}}$
 4. $Bob \rightarrow KDC\ B:$ $GTKT, TGT, \{T\}_{K_{B,KDCB}}$
 4. $Charlie \rightarrow KDC\ C:$ $GTKT, TGT, \{T\}_{K_{C,KDCC}}$
 5. $KDC\ B \rightarrow Bob:$ $GTKT = \{Alice@A, Group, K_{gk}, K_{help}^B\}_{K_B}, \{T\}_{K_{B,KDCB}}$
 5. $KDC\ C \rightarrow Charlie:$ $GTKT = \{Alice@A, Group, K_{gk}, K_{help}^C\}_{K_C}, \{T\}_{K_{C,KDCC}}$
-

Fig. 5. Kerberized multicast encryption protocol.

we discuss only tree-based multi-server multicast encryption schemes. To combine the protocols we note that:

- the key $K_{A,KDCA}$ in the crossrealm protocols and the key in the multicast encryption scheme shared between *Alice* and the server can be the same key;
- the key $K_{A,B}$ shared between *Alice* and *Bob* can be replaced by the group key that is shared during the execution of the (key-distribution phase of the) multicast encryption scheme;

Each of the proposed algorithms has the following properties:

- Allows for secure group key distribution to participants in the group. We consider a *join by invitation* where the initiator of the group contacts (invites) other entities to participate in a group communication.
- Achieves sender authentication in a group membership setting, which allows this algorithm to be used for the authenticated client join operation. To achieve sender authentication, the join algorithm described below can be easily modified.

7.1 Kerberized Multicast with Fake Tickets

We add the requirement that a key is shared among all centers in a preliminary stage. Note that before, analogously, a key was shared between KDC A and KDC B; this key was computed using public-key cryptography techniques (e.g., the two-party Diffie-Hellman key-exchange [4]). For the above protocol we can use any extension of these techniques to more than two parties (e.g., the n -party generalized Diffie-Hellman key-exchange [11]).

Figure 5 shows how *Alice* can send a message to the *Group*, which in this example includes *Bob* and *Charlie*, according to this protocol.

In step 1, *Alice* authenticates to her local KDC and requests a ticket for the multicast group, denoted *Group* in the Figure 5. *Alice* presents her Kerberos credentials: TGT and authenticator.

In step 2, *Alice*'s KDC creates a group key, K_{gk} , and a group ticket, $\{Alice@A, Group, K_{gk}\}_{K_{mk}}$. K_{mk} represents a key shared between the KDCs. KDC A also generates other multicast group keys, K_{help} for *Alice*. KDC A, as well as other KDCs, need to know membership information. One solution could be for *Alice* to include the information in the request. Then this membership information would be included in the group ticket; thus, other KDCs would have access to the membership list and be able to generate the necessary multicast keys for other members.

In step 3, *Alice* retrieves the group key from the reply and creates the authenticator using that key. She multicasts the message to the group. The message includes the group ticket and the corresponding authenticator.

In step 4, each of the members contacts their local KDCs as in the *fake* ticket protocol in order to validate and interpret the group ticket. All the requests are Kerberos authenticated. Each of the KDCs generates other multicast group keys from the group key if needed and converts *fake* tickets into regular tickets, by reencrypting the information with the appropriate keys.

In the last step, each of the participants receives the needed multicast keys and is assured that the message came from *Alice*.

Note that the protocol works regardless of whether the group members belong to the same realm as *Alice* or multiple realms, as in the example shown in Figure 5.

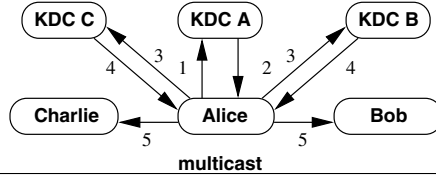
7.2 Integration with Original Crossrealm

The original crossrealm protocol fails to provide the same efficiency when used to secure group communication. Figure 6 shows how *Alice* can send a message to the *Group* according to this protocol. As in the previous example, the group consists of *Bob* and *Charlie*.

In step 1, *Alice* authenticates to her local KDC and request a ticket for the multicast group, simply named *Group* in the Figure. *Alice* presents her Kerberos credentials: TGT and authenticator. The request could include the membership list or *Alice* could include it in the request. From this list, the KDC infers which of the remote KDC are involved.

In step 2, KDC A first generates a group key, K_{gk} . Then it issues to *Alice* needed RTGTs. In our example, *Alice* receives remote ticket granting tickets for KDC B and KDC C. Each of the remote ticket granting tickets would need to include a newly created session key. Additionally, KDC A generates all other group keys required (which depend on the particular multicast encryption scheme used).

In step 3, *Alice* performs crossrealm authentication to all the remote KDCs. In each remote authentication request, *Alice* requests a ticket for a member of the



-
1. $Alice \rightarrow KDC\ A: Alice, Group, TGT = \{Alice, K_{A,KDCA}\}_{K_{KDCA}}, \{T\}_{K_{A,KDCA}}$
 2. $KDC\ A \rightarrow Alice: RTGT1 = \{Alice@A, K_{gk}, K_{A,KDCB}\}_{K_{KDCA,KDCB}},$
 $\{K_{A,KDCB}\}_{K_{A,KDCA}},$
 $RTGT2 = \{Alice@A, K_{gk}, K_{A,KDCC}\}_{K_{KDCA,KDCC}},$
 $\{K_{A,KDCC}\}_{K_{A,KDCA}}, \{K_{gk}, K_{help}^A, T\}_{K_{A,KDCA}}$
 3. $Alice \rightarrow KDC\ B: Alice@A, Bob@B, RTGT1, \{T\}_{K_{A,KDCB}}$
 3. $Alice \rightarrow KDC\ C: Alice@A, Charlie@C, RTGT2, \{T\}_{K_{A,KDCC}}$
 4. $KDC\ B \rightarrow Alice\ TKT1 = \{Alice@A, Bob@B, K_{gk}, K_{help}^B\}_{K_B}, \{T\}_{K_{A,KDCB}}$
 4. $KDC\ C \rightarrow Alice\ TKT2 = \{Alice@A, Charlie@C, K_{gk}, K_{help}^C\}_{K_C}, \{T\}_{K_{A,KDCC}}$
 5. $Alice \rightarrow Group: "Hi, I'm Alice", TKT1, TKT2 \{T\}_{K_{gk}}$
-

Fig. 6. Crossrealm multicast encryption protocol.

group that belongs to the realm. Each of the remote KDCs authenticates *Alice* by checking the RTGT and the corresponding authenticator. Each of the KDCs retrieves a group key from the RTGT and creates the requested ticket with that group key. For example a ticket to *Bob* would be $\{Alice@A, Bob@B, K_{gk}\}_{K_B}$. Also, for each member K_{help} multicast group keys are generated and included in the ticket.

In step 4, *Alice* awaits the replies from the remote KDCs. She checks the authenticators by validating the timestamps of the replies.

In step 5, *Alice* prepares the message to the group. The message includes all the tickets she received in the last step. *Alice* creates the authenticator, $\{T\}_{K_{gk}}$. She needs only one authenticator for all the tickets because the secret key inside each of the tickets is the same, namely K_{gk} . Each of the recipients can validate the ticket and the authenticator, then retrieve the group key from the ticket.

7.3 Discussion

Integration with the original crossrealm authentication protocol requires the client to contact all of the remote KDCs involved; i.e., $2 \cdot m$ message, where m is the number of KDCs. Furthermore, a join message multicast by *Alice* to all members is linear in size because it includes a ticket for each of the members. The use of *fake* tickets allows us to multicast a message with only a single ticket. *Alice* is not required to make any (potentially time consuming) requests. In terms of network delay, in the first protocol we have to account for $2 \cdot n$ Intranet messages. Due to the network latency of Internet message we assume that $n \cdot \alpha \leq m \cdot \beta$, where α is the network delay within an Intranet and β is the network delay on the Internet, while $m \leq n$.

8 Conclusion

In this paper we propose an efficient protocol for secure group communication via multicast, using Kerberos. We consider a practical distributed setting where users reside in different Intranets (realms) connected through the Internet. This setting is sufficiently general as almost any type of network can be characterized as described, and, yet, offers useful structure especially the presence of a trusted server inside every Intranet. We considered an extension to the original Kerberos protocol that enables crossrealm operations, identified its inefficiencies, and proposed a new paradigm for crossrealm Kerberos authentication that significantly decreases slow communications over the Internet, makes most operations local, and reduces the workload of the involved components. Furthermore, the paper describes new extensions of multicast encryption schemes to the case of multiple centers. Finally, we combine the new crossrealm protocol with those for the multi-server multicast encryption.

Acknowledgments. We thank the anonymous reviewers for their helpful comments. We also thank Peter Honeyman for his valuable comments.

References

1. K. Becker and U. Wille. Communication complexity of group key distribution. In *Proceedings of the 5th ACM Conference on Computer and Communication Security*, pages 1–6, San Francisco, CA, November 1998.
2. R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas. Multicast security: A taxonomy and efficient authentication. In *IEEE INFOCOMM*, 1999.
3. R. Canetti, T. Malkin, and K. Nissim. Efficient communication storage tradeoffs for multicast encryption. In *Proceedings of "Advances in Cryptology – EURO-CRYPT'99", Lecture Notes in Computer Science*, Springer Verlag, 1999.
4. W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transaction on Information Theory*, 22:644–654, November 1976.
5. M. Hur, B. Tung, T. Ryutov, C. Neuman, A. Medvinsky, G. Tsudik, and B. Sommerfeld. Public key cryptography for cross-realm authentication in kerberos, May 2001. Internet draft.
6. Y. Kim, A. Perrig, and G. Tsudik. Simple and fault-tolerant key agreement for dynamic collaborative groups. In *Proceedings of the 7th ACM Conference on Computer and Communication Security, CCS'00*, pages 235–244, November 2000.
7. F. MacWilliams and N. Sloane. *The theory of error-correcting codes*. Elsevier Science, 1977.
8. R. Needham and M. Shroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12):993 – 999, December 1978.
9. C. Neuman and T. Ts'o. Kerberos: an authentication service for computer networks. *IEEE Communications*, 32(9):33–38, September 1994.
10. A. Perrig. Efficient collaborative key management protocols for secure autonomous group communication. In *CryptTEC*, 1999.

11. M. Steiner, G. Tsudik, and M. Waidner. Diffie-hellman key distribution extended to groups. In *Proceedings of the 3rd ACM Conference on Computer and Communications in Security, CCS'96*, pages 31–37, March 1996.
12. D. Wallner, E. Harder, and R. Agee. Key management for multicast: Issues and architectures, June 1999. RFC 2627.
13. C. Wong, M. Gouda, and S. Lam. Secure group communication using key graphs. In *Proceedings of the ACM SIGCOMM'98*, pages 68–79, September 1998.

Suitability of a Classical Analysis Method for E-commerce Protocols

Sigrid Gürgens¹ and Javier Lopez²

¹ Fraunhofer - Institute for Secure Telecooperation SIT
guergens@sit.fraunhofer.de

² Computer Science Department, University of Malaga
jlm@lcc.uma.es

Abstract. We present the adaptation of our model for the validation of key distribution and authentication protocols to address specific needs of protocols for electronic commerce. The two models defer in both the threat scenario and in the formalization. We demonstrate the suitability of our adaptation by analyzing a specific version of the Internet Billing Server protocol introduced by Carnegie Mellon University. Our analysis shows that, while the security properties a key distribution or authentication protocol shall provide are well understood, it is often not clear what properties an electronic commerce protocol can or shall provide. Our methods rely on automatic theorem proving tools. Specifically, we used “Otter”, an automatic theorem proving software developed at Argonne National Laboratories.

Keywords: Security analysis, cryptographic protocol, automatic theorem proving, protocol validation, electronic commerce

1 Introduction

Cryptographic protocols play a key role in communication via open networks such as the Internet. These networks are insecure in the sense that any adversary with some technical background can monitor and alter messages interchanged by application users. Thus cryptographic protocols are used to provide the desired security properties. Messages are encrypted to ensure that only the intended recipient can understand them, digital signatures are applied to provide a proof that a message had been received, etc. All of these communication “services” can be provided as long as the cryptographic protocols themselves are correct and secure.

However, design of cryptographic protocols is a difficult and error-prone task, and many of these largely used protocols have been shown to contain flaws. For this reason the use of formal methods that allow for the verification/validation of such protocols in a systematic and formal way has received increasing attention. In the last ten to fifteen years, active areas of research have developed around the problems of:

- Developing design methodologies which yield cryptographic protocols for which security properties can be formally proven.
- Formal specification and verification/validation of cryptographic protocols.

An early paper which addresses the first of these issues is [4]. Here, it is argued that protocols should not be defined simply as communicating programs but rather as sequences of messages with verifiable properties; i.e. security proofs can not be based on unverifiable assumptions about how an opponent constructs its messages. As with much of the research work done in the area of cryptography, this research does not adopt the framework of formal language specifications. Some other work along the lines of specifying provably secure protocols includes that of Bellare and Rogaway [3], Shoup and Rubin [28] (which extends the model of Bellare and Rogaway to smartcard protocols), Bellare, Canetti and Krawczyk [2], and the work of Heintze and Tygar considering compositions of cryptoprotocols [11].

Another approach worth mentioning is [26]. Here agents are modeled by using communicating automata and security properties are defined and can be formally proven in terms of properties of abstract communication channels.

The second issue, formal specification and automatic verification or validation methods for cryptographic protocols, has developed into a field of its own. Partial overviews of this area can be found in [16], [14] and [22].

Most of the work in this field can be categorized as either development of logics for reasoning about security (so-called authentication logics) or as development of model checking tools. Both techniques aim at verifying security properties of formally specified protocols.

As regarding the first category, a seminal paper on logics of authentication is [6]. Work in this area has produced significant results in finding protocol flaws, but also appears to have limitations which will be hard to overcome within the paradigm.

Model checking, on the other hand, involves the definition of a state space (typically modelling the “knowledge” of different participants in a protocol) and transition rules which define both the protocol being analyzed, the network properties, and the capabilities of an enemy. Initial work in this area can be traced to [9].

Much has already been accomplished. A well-known software tool is the NRL Protocol Analyzer [15], which has recently been expanded to include some specification capabilities, thus being able to specify the security requirements of the SET protocol being used by Mastercard and Visa [18]. Other notable work includes Lowe’s use of the Failures Divergences Refinement Checker in CSP [13] and Schneider’s use of CSP [27]. Recently, the CSP model was extended to cover non-repudiation, a security requirement of e-commerce protocols [25]. To do so, the outside intruder was removed from the model and the protocol participants were given the ability to fake messages. However, it is not clear whether this model can be extended to malicious agents with the ability to intercept messages as well.

Also to mention are the model checking algorithms of Marrero, Clarke, and Jha [14]; and Paulson’s use of induction and the theorem prover Isabelle [22]. To the best of our knowledge, the model checking approach has been used almost exclusively for verification of cryptographic protocols.

Verification can be achieved efficiently if simplifying assumptions are made in order to obtain a sufficiently small state space. Verification tools which can handle infinite state spaces must simplify the notions of security and correctness to the point that proofs can be obtained using either induction or other logical means to reason about infinitely many states. Both methods have produced valuable insight into ways in which cryptographic protocols can fail.

The problem, however, is not only complicated but it is also evolving. The “classical” work has centered around proving security of key-distribution and authentication protocols, focusing on message properties such as key freshness, message confidentiality, and message origin authentication. Currently, with electronic commerce applications deployment, cryptographic protocols are being adapted for implementing commercial transactions.

This new generation of protocols imposes higher requirements on security issues. Secure electronic commerce functions such as transaction certification, electronic notary functions, operational performance, commerce disposal, anonymity, auditing, etc., are necessary for a successful deployment of electronic commerce transactions. These functions in turn produce requirements like identification of the sender and/or the receiver of a message, certification of delivery, confirmation of arrival, timestamping, detection of tampering, electronic storage of originality-guaranteed information, approval functions, and access control.

Consequently, protocols for electronic commerce scenarios are becoming much more complicated, and are more likely to contain errors. As a result, the complexity for their security analysis is increasing exponentially too. Moreover, electronic commerce transactions involve an increasing number of participants in one protocol run: Issuers, cardholders, merchants, acquirers, payment gateways and Certification Authorities (CAs). This again makes the security analysis problem harder to solve.

After Kailar’s approach to analyze accountability in electronic commerce protocols [12], there have been attempts to model other protocols such as Kerberos [1], TLS/SSL [23], Cybercash coin-exchange [5], and, as already mentioned, SET [18].

Already existing analysis techniques can be successfully adapted, and hence, applied to electronic commerce protocols (see [25]). In this paper we show how the method we have developed for the analysis of classical protocols can be adapted to cover specific needs of e-commerce protocols.

The rest of the paper is organized as follows: Section 2 describes the communication and attack model we developed for the analysis of protocols related to key distribution and authentication (which we will call “classical protocols” throughout this paper). Section 3 illustrates the way the theorem prover Otter is used for protocol analysis purposes, following criteria established in the previous section. In section 4 we present a new type of attack and discuss some known

attacks, in section 5 we explain how the model described in section 3 is adapted for the analysis of electronic commerce protocols. As an example, section 6 and 7 describe the IBS payment protocol and the results of our analysis, respectively. Section 8 concludes the paper.

2 The Communication and Attack Model

The security analysis of protocols does not deal with weaknesses of the encryption and decryption functions used. In what follows we assume perfect encryption in the following sense:

1. Messages encrypted using a function f and a secret key K can only be decrypted with the inverse function and key f^{-1}, K^{-1} .
2. A key can not be guessed.
3. For the generation of a ciphertext $\{m\}_K$, it is necessary to know both m and K .


While the first two items describe generally accepted properties of encryption functions, the last one does not hold in general. Actually, to date no proof is known for a cryptalgorithm to provide this property.

All communication is assumed to occur over insecure communication channels. We model these (in accordance with the Dolev-Yao attack model) by assuming that there is one more agent E that intercepts all messages sent by others. After intercepting, E can change the message to anything it can compute. This includes changing the destination of the message and the supposed identity of the sender. Furthermore, E can be considered a valid agent, thus may or may not also share a symmetric key K_{ES} with S and/or own a key pair (SK_E, PK_E) for an asymmetric algorithm, and can be allowed to initiate communication either with other agents or S . Our model leaves these decisions open to be fixed for a particular analysis run.

E intercepts all messages and the agents and S only receive messages sent by E . What E can send depends on what it is able to generate, and this in turn depends on what E knows. The knowledge of E can be recursively defined as follows:

1. Depending on the decision for a particular analysis run, E may or may not know the names of the agents.
2. E knows the keys it owns.
3. E knows its own random numbers.
4. E knows every message it has received.
5. E knows every part of a message received (where a ciphertext is considered one message part).
6. E knows everything it can generate by enciphering data it knows using data it knows as a key.
7. E knows every plaintext it can generate by deciphering a ciphertext it knows, provided it knows the inverse of both the key and the encryption function used as well.

8. E knows every concatenation of data it knows.
9. E knows the origin and destination of every message.
10. At every instance of the protocol run, E knows the "state" of every agent, i.e. E knows the next protocol step they will perform, the structure of a message necessary to be accepted, in principle what they will send after having received (and accepted) a message, etc.

At every state in a possible run, after having received an agent's message, E has the possibility to forward this message or to send whatever message it can generate to one of the agents or S . Obviously, the state space will grow enormously. Thus, we must limit the concept of knowledge if the goal is to produce a validation tool which can be used in practice. Therefore we restrict the definition in various ways in order to make E 's knowledge set finite. For example, if the protocol to be analyzed uses a public key algorithm, we do not allow E to generate ciphertexts using data as a key which is neither the private nor the public part of a key pair. We do this on an ad-hoc basis. This problem needs further exploration. 

Another design decision concerns the question of whether or not messages contain types, i.e. whether a recipient of a message can distinguish, say, a random number from a cryptographic key. There is an open discussion in the research community on whether protocol flaws resulting from type confusion are relevant. We believe that this question can not be decided a priori but is dependant on the message format of the particular application that the protocol is used in. It may be advantageous to know that a particular type check is not needed in order to achieve the desired security properties (by skipping unnecessary checks, performance can be improved), but it is absolutely necessary to know that a particular check is needed to avoid an attack in those applications that do not perform type checks by default. Consequently, our analysis model does provide the possibility to add types to messages, but for the analysis of classical protocols we assume that messages are not typed.

3 Formalization of Protocol Instantiations

As the title of this section suggests, we make a distinction between a protocol and a "protocol instantiation". We deem this a necessary distinction because protocols in the literature are often under-specified. A protocol may be implemented in any of several ways by the applications programmer.

The theorem prover Otter we used for protocol analysis is based on first order logic. Thus for formalizing a protocol, the environment properties and possible state transitions we use first order functions, predicates, implications and the usual connectives. At any step of a protocol run, predicates like

$state(xevents, xEknows, \dots, xwho, xto, xmess, \dots)$

¹ We note, however, that leaving this part of our model open allows for a very general tool. This is in part responsible for the success of the tool, as documented in later sections.

describe a state of the system: each predicate contains the list $xevents$ of actions already performed, the list $xEknows$ of data known by E , and additionally, for each agent and each run it assumes to be taking part in, components $xagent, xrun, xstep, xrandom, xkey$, etc. that describe the states of the agent in the particular run (e.g. data like session keys, nonces, etc., the agent has stored so far for later use, and the next step it will perform). In general the predicates can include as many agents' states and protocol runs as desired, but so far we have restricted this information to at most two agents, a server and the attacker in the context of two interleaving protocol runs. Furthermore, the predicates contain, among other data used to direct the search, the agent $xwho$ which sent the current message, the intended recipient xto and the message $xmess$ sent. Messages in our model are abstract objects, ordered tuples (m_1, m_2, \dots, m_r) of concatenated messages m_i which are viewed at an abstract level where the agents can distinguish between different parts.

Using these predicates we write formulas which describe possible state transitions. For the first step of the Needham-Schroeder Protocol (where agent A sends to the key server S the message A, B, R_A , see section 4), the action of S when receiving the message can be formalized as follows:

$$\begin{aligned}
 & send(xevents, xEknows, \dots, xwho, xto, xmess, \dots) \\
 & \wedge xto = S \\
 & \wedge xwho = E \\
 & \wedge is_agent(el(1, xmess)) \wedge is_agent(el(2, xmess)) \\
 & \rightarrow \\
 & send([\dots | xevents], xEknows, \dots, S, el(1, xmess), \\
 & [script(key(el(1, xmess), S), [el(3, xmess), el(2, xmess)], new_key, \\
 & script(key(el(2, xmess), S), [new_key, el(1, xmess)]))], \dots)
 \end{aligned}$$

where $[\dots | xevents]$ includes the current send action, $el(k, xmess)$ returns the k th block of the message $xmess$ just received, and new_key stands for a number uniquely generated for each protocol run. By checking that the message was sent by E ($xwho = E$) we model the fact that agents only receive messages sent by E .

Whenever a message is intercepted by E (indicated by a formula $send(\dots) \wedge xwho \neq E$), an internal procedure adds all that E can learn from the message to the knowledge list $xEknows$. The result of this procedure is a predicate $added_all_knows(\dots)$. Implications $added_all_knows(\dots) \rightarrow send(\dots, xwho, \dots, message1, \dots) \wedge send(\dots, xwho, \dots, message2, \dots) \wedge \dots$ with $xwho = E$ formalize that after having intercepted a particular message, E has several possibilities to proceed, i.e. each of the predicates $send(\dots, message, \dots)$ constitutes a possible continuation of the protocol run. The messages E sends to the agents are constructed using the basic data in $xEknows$ and the specification of E 's knowledge as well as some restrictions as discussed in the previous section.

Symmetric keys shared by agents X and Y are denoted by $key(X, Y)$, $key(X, pub)$ denotes the public key and $key(X, priv)$ the private key of agent

X . For being able (if necessary) to distinguish between different cryptographic algorithms, we use $script(k, m)$ to formalise the encryption of message m with key k and the symmetric algorithm $script$, $sdecrypt(k, m)$ formalizes the decryption of message m . Similarly an asymmetric algorithm being used is indicated by $acrypt(k, m)$ and a digital signature by $sig(k, m)$.

We use equations to describe the properties of the encryption and decryption functions, keys, etc., used in the protocol. For example, the equation

$$sdecrypt(x, script(x, y)) = y$$

formalizes symmetric encryption and decryption functions where every key is its own inverse. These equations are used as demodulators in the deduction process.

The above formulas constitute the set of axioms describing the particular protocol instantiation to be analyzed. An additional set of formulas, the so-called "set of support", includes formulas that are specific for a particular analysis. In particular, we use a predicate describing the initial state of a protocol run (the agent that starts the run, with whom it wants to communicate, the initial knowledge of the intruder E , etc.) and a formula describing a security goal. An example of such a security goal is "if agent A believes it shares a session key with agent B , then this key is not known by E ", formalized by $-(state(\dots) \wedge xAstep = finished \wedge xAgent = B \wedge in_list(xAkey, xEknows))$

Using these two formula sets and logical inference rules (specifically, we make extensive use of hyper resolution), Otter derives new valid formulas which correspond to possible states of a protocol run. The process stops if either no more formulas can be deduced or Otter finds a contradiction which involves the formula describing the security goal. When having proved a contradiction, Otter lists all formulas involved in the proof. An attack on the protocol can be easily deduced from this list.

Otter has a number of facilities for directing the search, for a detailed description see [30].

4 Known and New Attacks

We used Otter to analyze a number of protocols. As an example, we show the analysis of the well-known symmetric Needham-Schroeder key distribution protocol [19]. The protocol comprises the following steps:

1. $A \longrightarrow S : A, B, R_A$
2. $S \longrightarrow A : \{R_A, B, K_{AB}, \{K_{AB}, A\}_{K_{BS}}\}_{K_{AS}}$
3. $A \longrightarrow B : \{K_{AB}, A\}_{K_{BS}}$
4. $B \longrightarrow A : \{R_B\}_{K_{AB}}$
5. $A \longrightarrow B : \{R_B - 1\}_{K_{AB}}$

S denotes a trusted key distribution server, $\{m\}_{K_{XY}}$ denotes the encryption of the message m using key K_{XY} (shared by agents X and Y), and R_X denotes a random number generated by agent X for a protocol run.

Whenever we model a protocol we are confronted with the problem that usually the checks performed by the parties upon receiving the messages are not specified completely. This means that we must place ourselves in the situation of the implementer and simply encode the obvious checks. For example, consider step 3 of the previous protocol.

Here is a case where we have to decide what B 's checks are. We did the obvious: (a) B decrypts using K_{BS} ; (b) B checks that the second component of the decrypted message is some valid agent's name; and (c) B assumes, temporarily, that the first component of the decrypted message is the key to be shared with this agent.

However, Otter determined that this was a fatal flaw in the implementation. We call the type of attack found the "arity" attack, since it depends on an agent not checking the number of message blocks embedded in a cyphertext. The arity attack found by Otter works as follows: in the first message E impersonates B (denoted by E_B) and sends to S :

1. $E_B \longrightarrow S : B, A, R_E$

Then, S answers according to the protocol description and sends the following message:

2. $S \longrightarrow B : \{R_E, A, K_{AB}, \{K_{AB}, B\}_{K_{AS}}\}_{K_{BS}}$

The message is intercepted by E and then passed on to B , and B interprets it as being the third message of a new protocol run. Since this message passes the checks described above, B takes R_E to be a new session key to be shared with A and the last two protocol steps are as follows:

4. $B \longrightarrow E_A : \{R_B\}_{R_E}$
5. $E_A \longrightarrow B : \{R_B - 1\}_{R_E}$

where the last message can easily be generated by E since it knows its random number sent in the first protocol step (see [10] for a formalization of B 's actions).

It is interesting to note that an applications programmer is quite likely to not include an "arity-check" in the code. This is because modular programming naturally leads to the coding of routines of the type

$$get(cyphertext, key, i)$$

which decodes *cyphertext* with *key* and returns the i^{th} element of the plaintext.

By including the arity check in B 's actions when receiving message 3, we analyzed a different instantiation of the same protocol. This time Otter found the well-known Denning-Sacco attack [7], where the third message of an old protocol run is replayed to B and the attack run ends with B and the intruder E sharing an old session key that possibly was broken by E .

It seems that many protocols may be subject to the "arity" attack: our methods found that this attack is also possible on the Otway-Rees protocol (see [10] for a detailed description of the protocol and its analysis) and on the Yahalom protocol, as described in [6].

Among the known attacks which were also found by our methods are the Lowe ([13]) and the Meadows [17] attacks on the Needham-Schroeder public key protocol and the Simmons attack on the TMN protocol (see [29]). For the latter we relaxed the assumption of perfect encryption and modeled the homomorphic

property of crypt algorithms like RSA [24] that allow to generate a ciphertext by multiplication.

Furthermore, we analyzed an authentication protocol for smartcards with a digital signature application standardized by the German standardization organisation DIN ([8]) and found the particular version of the standard to be subject to some interpretations which would make it insecure.

5 The Extended Model and Its Formalization

As already pointed out in the first section, e-commerce protocols are much more complex than traditional cryptographic protocols with respect to security requirements and threats. While in traditional protocols all agents are assumed to act correctly and the only threat is being posed by a hostile environment, in e-commerce protocols we have a situation where agents do not necessarily trust each other. A customer in a payment system might want to gain advantage of the system by not paying the agreed amount of money, a service provider may try to get away with not sending the service. So it is not so much the abilities of an outside intruder we are interested in but the abilities of the agents themselves that are taking part in a protocol.

As to security properties the protocols shall provide, we still have to deal with confidentiality and authenticity, but furthermore there are new properties that pose different questions. So besides the fact that protocol flaws may arise because of type confusion, we are interested in questions like "Is it possible for the client to receive the service without ever delivering the payment?", in which case we search for a system state where the service provider did deliver the service, the client owns whatever the protocol provides as a proof of having paid, but never did.

It is clear that the difference in security threats between classical and e-commerce protocols induces differences in the formalization of protocol and attack scenario. For the rest of the paper, we will refer to the analysis model for classical cryptoprotocols as the "classical model" and to that for e-commerce protocols as to the "e-commerce model".

Our communication and attack model for the analysis of e-commerce protocols is not the traditional Dolev-Yao model, where agents are assumed to act honestly and where an intruder is in the center of communication, intercepts every message and can send whatever message it is able to generate. Our model does contain an intruder, but additionally we assume that any number of agents participating in the system can be dishonest at any time. Regarding this, our model is more flexible than the model of mobile adversary introduced in [20], since here it is assumed that at most k out of n agents of the system can be corrupted. The capability of agents to generate faked messages is based on their knowledge, which is derived along the "rules" given in section 2. Furthermore, we also allow a situation where dishonest agents are capable of intercepting messages as well. This may or may not be relevant, depending on the environment in which to use the protocol.

In general our model allows an unrestricted number of dishonest agents, but so far we have restricted analysis runs to at most two. In the case of two dishonest agents, we assume that they cooperate, i.e. exchange whatever knowledge they have, except that they do not make available to each other their private keys. It is very unlikely that one analysis run with two dishonest agents acting independently gives more results than two analysis runs with one dishonest agent each.

As in the classical model, we are not concerned with the security of the algorithms that are being used in the protocols, i.e. we again assume perfect encryption. But in contrast to the analysis of classical protocols where we assume that agents are not able to distinguish between different types, in the e-commerce model we add types to the messages. Thus the agents can be modeled as being able or not being able to distinguish between different types of messages. This allows the analysis to concentrate on issues specific to e-commerce protocols rather than searching for protocol flaws caused by type confusion, but it keeps the search for type confusion based flaws possible.

Consequently, in the analysis runs performed so far we allowed dishonest agents only to send messages in the correct format, e.g. a message identification number will not be sent as a price, a signature will not be used as a random number, etc.

For the formalization of e-commerce protocols we had to make a few changes to predicates and formulas, the most important ones are listed below.

- The predicates additionally contain a list of agents that act dishonestly and a list of agents that cooperate (which allows for future extension of the number of cooperating agents). Furthermore, agents own a list of keys and a list for keeping information they extracted from messages not intended for them.
- In the classical cryptoprotocols we analysed so far there was no need to model an agent sending two messages subsequently. However, in e-commerce protocols the situation may be different, in particular when allowing for cooperating malicious agents whose actions are not fixed by the protocol description. Thus we introduced a "dummy message": If agent X sends first a message to agent Y and then a message to agent Z , we model this by having Y answer to X with *dummy*, after which X can proceed with sending the second message to Z .
- We model the interception of messages by dishonest agents principally in the same way as in the classical model: Whenever a message is sent by an honest agent ($send(...) \wedge \neg in_list(xwho, xbadguys)$), an internal procedure adds all that the dishonest agents can learn from the message to the respective knowledge list $xagentknows$. If there are cooperating agents, their knowledge lists are exchanged. Finally for each of the dishonest agents the messages to be sent are constructed, on the basis of the agent's knowledge and keys he owns. The messages are then sent to those agents that will accept the message with respect to the format (e.g. an order will not be sent to a client, but only to a service provider). If we want to analyse a protocol in an environment where agents can not intercept messages, we skip the internal

procedure of knowledge extraction and connect the reception of messages directly with the generation of faked messages by way of demodulators such as $in_list(xto, xbadguys) \rightarrow send(\dots) = construct_messages(\dots)$.

6 The Internet Billing Server Protocol

Using the above described model, we analyzed the Internet Billing Server Protocol (IBS protocol) developed by Carnegie Mellon University and described in [21]. Actually, [21] includes several different versions of protocol templates where the precise format of messages is left open. We chose the version that uses only asymmetric algorithms and made those changes we deemed necessary for a reasonable analysis.

The protocols were designed to enable sales of goods to be delivered over the network. There are three different types of agents: service providers, users (those agents which buy the goods) and the Billing Server, trusted by all agents, which handles the money transfer. Both service provider SP and user $User$ register with the billing server BS , which means in particular that accounts are opened and key pairs are issued by BS .

The protocol assumes that a signature algorithm with message recovery is used (an algorithm that allows to decipher a signature in order to get the plain-text again), $\{message\}_{SK_X}$ denotes the message signed with the private key of agent X . In the following, we describe our version of the IBS protocol. It consists of two phases, the first of which is price delivery, where the user asks some service provider for a price and the service provider answers with a price.

1. $User \rightarrow SP : \{ID, request, servicename\}_{SK_{User}}$
2. $SP \rightarrow User : \{ID, price\}_{SK_{SP}}$

The second phase is the service provision and payment phase:

3. $User \rightarrow SP : \{\{ID, price\}_{SK_{SP}}, ID, price\}_{SK_{User}}$
4. $SP \rightarrow BS : \{\{\{ID, price\}_{SK_{SP}}, ID, price\}_{SK_{User}}\}_{SK_{BS}}$
5. $BS \rightarrow SP : \{ID, authorization\}_{SK_{BS}}$
6. $SP \rightarrow User : \{ID, service\}_{SK_{SP}}$
7. $User \rightarrow SP : \{ID, ackn\}_{SK_{User}}$
8. $SP \rightarrow log : \{\{ID, ackn\}_{SK_{User}}\}_{SK_{SP}}$
9. $SP \rightarrow BS : log$

First, in step 3, the user orders the service. In step 4 the service provider asks the Billing Server for authorization of this service, which means in particular that the Billing Server checks that the user's account provides enough money. If so, the Billing Server transfers the price for the service from the user's to the service provider's account and keeps a record with all data relevant for this transaction. Then he authorizes the service (step 5) and the service is delivered (step 6). The user's last action is to send a message acknowledging that he received the service. The service provider collects all acknowledgement messages in a log file which is sent to the Billing Server in off-peak hours (step 9). When receiving the

log file, the Billing Server checks that the acknowledgement messages in the log file match the previously performed money transfers.

We added a message ID to all messages of the original version, and the constant “*request*” and the variable *servicename* to the first message, taking into account that a price will depend on the time it was given, the user it was given to, the service, and the service provider, that an authorization refers to a specific amount of money to be paid by a specific user to a specific service provider for a specific service, etc. The identification number both serves as a time stamp (it is issued by the user and unique for each run) and connects the messages of one run.

Some general assumptions were made in [21]:

1. In case of dispute, the user is always right. In consequence, if the service provider tries to cheat and the user complains about the charge, the service provider will always lose. On the other hand, the service provider can protect himself against malicious users by refusing access to the system.
2. All agents have access to a verifiable source of public keys, none of the keys gets compromised.
3. To secure the protocol against replay attacks etc., “time stamps, nonces, and other well documented means” can be used.

A number of security requirements the protocol imposes are explained in [21], some of which we list below.

1. The service provider can only be held responsible for those prices he indeed offered.
2. The user must be sure he will only be charged for services he received for the price mutually agreed on.
3. The service provider must be sure that the user is not able to receive a service and deny the receipt.

Clearly, the protocol does allow the user to deny having ever received the service: he can just refuse to send the acknowledgement message. According to [21], the protocol is designed for services with small value only, thus a service provider can always deny a user that has refused to send acknowledgement messages further access to the system.

6.1 Our Assumptions and Formalization

To model this particular version of the IBS protocol, we used the following assumptions:

1. The system consists of two Users *User1*, *User2*, two service providers *SP1*, *SP2*, the Billing Server *BS* and an outside intruder *E*. The Billing Server is the only agent that always behaves according to the protocol description. Furthermore, we have two services *service1*, *service2* with respective service names and different prices to start with for the service providers. The price is incremented after having been used. (However, we can use a initial state where both service providers use the same price as a starting point.)

2. All agents own valid key pairs and the public key of all other agents.
3. All predicates contain, for each user and each service provider, an additional component *xagentmemory*. For each protocol run an agent is engaged in we have an entry in this component that contains the *ID*, the user / service provider he is communicating with, the service that is being negotiated, the price given and the state the agent is in. Equivalently, the Billing Server holds a list of authorization requests (*xauthrequ*), each of which contains *ID*, *price*, *User* and *SP* being engaged in the particular negotiation. Additionally, *BS* holds a list with the accounts of users and service providers.
4. Although in [21] it is not explicitly said so, we assume that all signatures are checked. Since the protocol does not provide information on who signed the messages, we added a further component *xdetmessages* to the predicates that includes the *ID*, the message type (i.e. *message1*, *message2*, etc.) and the signers of the message, starting with the outermost signature, the next inner one, then the innermost signature (if there are that many). This allows to identify the signer of the first message (a service provider can not know who has sent a particular price request), and to identify which run the message belongs to by use of the *ID*.

Whenever in the recipient's *xagentmemory* there exists already an entry with the *ID* given in *xdetmessages*, the public keys to be used for signature verification are determined by using the public keys of the user and service provider, respectively, given in this particular entry. This models the fact that a user will only accept a service provided by the service provider he ordered the service from, that a service provider will only accept an order that was sent by the user he offered the specific price to, etc. However, there may be situations where only the integrity of the message needs to be ensured and the actual signer is not of interest. For example, a user might not be interested in knowing who actually sent the mpeg file he ordered. To model this we can determine the public keys to be used for signature verification by using the agents given in *xdetmessages*.

7 Our Analysis of the IBS Protocol

As already pointed out in the previous section, there are a number of security requirements to be met. In the following we will list a few of these requirements and the formula that models the respective property (thus the formula to be found by Otter for finding a contradiction and therefore an attack).

1. The service provider can only be held responsible for those prices he indeed offered. This can be formalized with the formula $\neg(\text{state}(\dots) \wedge \text{give_price}(xSP, xID) > \text{give_price}(xauthrequ, xID))$.
2. The user must be sure he will only be charged for services he received at the price mutually agreed on. These are actually two requirements that can be captured by $\neg(\text{state}(\dots) \wedge \neg \text{service_delivery}(xauthrequ, xevents))$ and $\neg(\text{state}(\dots) \wedge \text{give_price}(xUser, xID) > \text{give_price}(xauthrequ, xID))$.

3. The service provider must be sure that the user is not able to receive a service and deny the receipt: $\neg(\text{state}(\dots) \wedge \text{denial_ackn}(xUsermemory, xSPmemory))$.

The values of the predicates *give_price*, *denial_ackn*, etc., are determined by means of conditional demodulation.

For the analysis, in general we assumed that all possible checks are performed. This includes that the service provider checks, receiving message 3, that the price given in his own signature (which is part of the message signed by the user) is the same as the one signed by the user; that the Billing Server checks that the two signatures of the service provider in the request for authorization (inner and outer) are performed by the same service provider, etc. This also includes that the Billing Server checks, when receiving an authorization request, that he has not yet received a request with the *ID* given.

The first protocol flaw we found is the one already stated in [21]: By modeling a malicious user that can stop the protocol run at any given point, we found (which is not surprising) a state in which the user has stopped the run and the service provider has not received an acknowledgement message.

We then disabled the user's ability to stop runs and let the analysis run with a malicious user and a malicious service provider. Now Otter found that the protocol can be run without performing step 6, i.e. without service delivery, and the Billing Server still accepts the log file containing the user's acknowledgement message. Thus the protocol does allow a situation where the user is charged without having received a service. One may argue that this can not be considered an attack, since it is only possible with the active cooperation of the user, and thus is no violation of the security requirement that a user may only be charged for services actually received. In many cases this will be right. However, our analysis reveals the fact that the messages received by the Billing Server do not constitute a proof that a service was delivered but a proof that the user agrees on being charged a specific amount of money. There may be environments where it matters whether or not the Billing Server can be used for transferring money from one account to another without a service being involved, environments where any kind of "money laundry" has to be avoided.

Our analysis also shows that care must be taken when specifying the security requirements. One way to formalize security requirements precisely is by way of system states that describe a violation of the desired property. In an environment where the above described protocol run is unacceptable, the formula $\neg(\text{state}(\dots) \wedge \neg \text{service_delivery}(xauthrequ, xevents))$ captures the fact that **no** user shall be charged without having received a service. If "money laundry" is of no importance, the requirement can be relaxed by adding $\wedge \neg \text{in_list}(xuser, xbadguys)$.

8 Conclusions

As is well known, there does not exist an algorithm which can infallibly decide whether a theorem is true or not. Thus the use of Otter, and per force of any

theorem proving tool, is an art as well as a science. The learning curve for using Otter is quite long, which is a disadvantage. Furthermore, one can easily write Otter input which would simply take too long to find a protocol failure even if one exists and is in the search path of Otter's heuristic. Thus, using this kind of tool involves the usual decisions regarding the search path to be followed.

Nevertheless, our approach has shown itself to be a powerful and flexible way of analyzing protocols. By making all of the agents' actions explicit, our methods reveal implicit assumptions that are not met by the protocol being analyzed. In particular, our methods found protocol weaknesses not previously found by other formal methods. On the other hand, as we are concerned with protocol validation rather than verification, if our analysis does not find an attack, we can not conclude that the protocol is secure in general. All we know is that under certain assumptions concerning the security properties of the cryptoalgorithms used and the abilities of malicious agents, a certain state is unreachable.

Furthermore, we have successfully adapted our methods to address the specific needs of e-commerce protocol analysis. The new model for the analysis of e-commerce protocols is flexible enough to be used for different threat scenarios with respect to possibilities of malicious agents. Our analysis of a specific version of the IBS protocol both shows our methods to be applicable to e-commerce protocols and emphasises that care must be taken when identifying the security requirements of such a protocol. A way to formalize the desired properties is to use formulas that describe a system state where the property in question is violated. Future work will include the formalization of more security properties relevant for e-commerce protocols and the application of our methods to other e-commerce protocols.

References

1. G. Bella and L.C. Paulson. Kerberos version iv: Inductive analysis of the secrecy goals. In *5th European Symposium on Research in Computer Security*, Lecture Notes in Computer Science, pages 361–375. Springer-Verlag, 1998.
2. M. Bellare, R. Canetti, and H. Krawczyk. A Modular Approach to the Design and Analysis of Authentication and Key Exchange Protocols. In *Annual Symposium on the Theory of Computing*. ACM, 1998.
3. M. Bellare and P. Rogaway. Provably secure session key distribution - the three party case. In *Annual Symposium on the Theory of Computing*, pages 57–66. ACM, 1995.
4. R. Berger, S. Kannan, and R. Peralta. A framework for the study of cryptographic protocols. In *Advances in Cryptology – CRYPTO '95*, Lecture Notes in Computer Science, pages 87–103. Springer-Verlag, 1985.
5. S. Brackin. Automatically detecting authentication limitations in commercial security protocols. In *Proc. of the 22nd National Conference on Information Systems Security*, October 1999.
6. M. Burrows, M. Abadi, and R. Needham. A Logic of Authentication. Report 39, Digital Systems Research Center, Palo Alto, California, Feb 1989.
7. D. Denning and G. Sacco. Timestamps in key distribution protocols. *Communications of the ACM*, 24:533–536, 1982.

8. DIN NI-17.4. *Spezifikation der Schnittstelle zu Chipkarten mit Digitaler Signatur-Anwendung / Funktion nach SigG und SigV, Version 1.0 (Draft)*, November 1998.
9. D. Dolev and A. Yao. On the security of public-key protocols. *IEEE Transactions on Information Theory*, 29:198–208, 1983.
10. S. Gürgens and R. Peralta. Efficient Automated Testing of Cryptographic Protocols. Technical report, GMD German National Research Center for Information Technology, 1998.
11. N. Heintze and J.D. Tygar. A Model for Secure Protocols and their Compositions. In *1994 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 2–13. IEEE Computer Society Press, May 1994.
12. R. Kailar. Accountability in Electronic Commerce Protocols. In *IEEE Transactions on Software Engineering*, volume 22, pages 313–328. IEEE, 1996.
13. G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using CSP and FDR. In *Second International Workshop, TACAS '96*, volume 1055 of *LNCS*, pages 147–166. SV, 1996.
14. W. Marrero, E. M. Clarke, and S. Jha. A Model Checker for Authentication Protocols. In *DIMACS Workshop on Cryptographic Protocol Design and Verification*, <http://dimacs.rutgers.edu/Workshops/Security/>, 1997.
15. C. Meadows. A system for the specification and verification of key management protocols. In *IEEE Symposium on Security and Privacy*, pages 182–195. IEEE Computer Society Press, New York, 1991.
16. C. Meadows. Formal Verification of Cryptographic Protocols: A Survey. In *Advances in Cryptology - Asiacrypt '94*, volume 917 of *LNCS*, pages 133 – 150. SV, 1995.
17. C. Meadows. Analyzing the Needham-Schroeder Public Key Protocol: A Comparison of Two Approaches. In *Proceedings of ESORICS*, Naval Research Laboratory, 1996. Springer.
18. C. Meadows and P. Syverson. A formal specification of requirements for payment transactions in the SET protocol. In *Proceedings of Financial Cryptography*, 1998.
19. R. Needham and M. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, pages 993–999, 1978.
20. R. Ostrovsky and M. Yung. How to withstand mobile virus attacks. In *Proceedings of PODC*, pages 51–59, 1991.
21. K. O'Toole. The Internet Billing Server - Transaction Protocol Alternatives. Technical Report INI TR 1994-1, Carnegie Mellon University, Information Networking Institute, 1994.
22. L. C. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6:85–128, 1998.
23. L. C. Paulson. Inductive Analysis of the Internet Protocol TLS. *ACM Trans. on Information and System Security*, 2(3):332–351, 1999.
24. R. L. Rivest, A. Shamir, and L. A. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.
25. B. Roscoe, P. Ryan, S. Schneider, M. Goldsmith, and G. Lowe. *The modelling and Analysis of Security Protocols*. Addison Wesley, 2000.
26. C. Rudolph. A Formal Model for Systematic Design of Key Establishment Protocols. In *Information Security and Privacy (ACISP 98)*, Lecture Notes in Computer Science. Springer Verlag, 1998.
27. S. Schneider. Verifying authentication protocols with CSP. In *IEEE Computer Security Foundations Workshop*. IEEE, 1997.

28. V. Shoup and A. Rubin. Session key distribution using smart card. In *Advances in Cryptology - EUROCRYPT '96*, volume 1070 of *LNCS*, pages 321–331. SV, 1996.
29. M. Tatebayashi, N. Matsuzaki, and D. Newman. Key Distribution Protocol for Digital Mobile Communication Systems. In G. Brassard, editor, *Advances in Cryptology - CRYPTO '89*, volume 435 of *LNCS*, pages 324–333. SV, 1991.
30. L. Wos, R. Overbeek, E. Lusk, and J. Boyle. *Automated Reasoning - Introduction and Applications*. McGraw-Hill, Inc., 1992.

Hyppocrates

A New Proactive Password Checker

Carlo Blundo, Paolo D'Arco, Alfredo De Santis, and Clemente Galdi

Dipartimento di Informatica ed Applicazioni
Università di Salerno, 84081 Baronissi (SA), Italy
{carblu, paodar, ads, clegal}@dia.unisa.it

Abstract. In this paper we propose a new *proactive password checker*, a program which prevents the choice of easy-to-guess passwords. The checker uses a decision tree which has been set up applying the Minimum Description Length Principle and a Pessimistic Pruning Technique to refine its predictive power. Experimental results show a substantial improvement in performances of this checker with respect to previous proposals. Moreover, the system is user-friendly and can be adapted to a per-site policy by the system administrator.

1 Introduction

The rapid growth on the Internet of software applications with high security requirements has increased the need of authentication tools and identification protocols for checking the access to shared resources. Among these ones, the old password approach is still the more frequently used for its simplicity.

When a user wants to identify himself to a system, during an interactive checking identification protocol, he types on a keyboard a secret word, the *password*, as an answer to an identification query. The system usually stores in a database some information encrypted using the password as a key. When it receives the password, it re-computes the encrypted information and compares this with the stored one. If they match, the access is granted; otherwise it is refused. This mechanism is for example used in Unix-like operating systems [20].

Of course, the scheme is secure if the user keeps secret his own password. However, it is not enough. In the following sections we will try to explain why the secrecy of the password does not guarantee access security looking back at previous works on this subject and describing our solution.

1.1 Password Choice: The Issue of Security

Network security can be attempted in various ways. The violations can be done by physical users as well as by malicious softwares. *Unauthorized accesses* or *illegal uses* of system resources, *virus*, *worms*, and *Trojan horses* are all examples of this multispects issue. A detailed survey of these different topics can be found in [20].

In this paper, we want to focus our attention on *access control*. Here, unauthorized users, called *hackers or crackers*, try to log on a machine. Once logged in, these intruders can behave in different ways [9].

The main way used by the hackers to log on to a machine is by mean of a user's password. This is possible not only when the user accidentally discloses his password but even when it is "easy" to guess. Indeed, the users frequently choose passwords that are pronounceable and simply to remember. Experimental results, reported in [12], show that many users choose short passwords or trivial ones, and passwords that belongs to well know dictionaries.

Thus an hacker, having a copy of the file containing the encrypted information used by the system during the access control process, can try off-line to "guess" one of the passwords in order to match an item of the file. Indeed, if such a match is found, the guessed password enables him to gain access to the system. Since the set of passwords that the users effectively choose is small, it can be exhaustively checked in a reasonable time. Excellent softwares to accomplish this task have been developed in the recent years and are available on-line [13].

Moreover, the problem of choosing good passwords is not restricted to access control of network systems. Passwords are often used to protect *private information*. Simple examples can be the use of a password to protect cryptographic keys, stored on the local disk of the user or password-authenticated key agreement schemes [8][11].

A more complex example can be the use of a password in systems like Kerberos described in [15]. Even in these cases it is needed to use good passwords.

Therefore, to increase the security level of a password based system, we have to find *a method* that reduces the efficacy of the *exhaustive search attacks*. This goal can be achieved if users are not allowed to choose easy to guess passwords. Indeed, if the password can be any element of a given set of big size, then an exhaustive search cannot be tried in a reasonable time.

1.2 Previous Work

The problem of using good passwords has been studied in several papers. Bishop in [45] has deeply analyzed this problem. He has given a series of hints about password subsets and easily transformations on words of a dictionary, usually applied by the users to generate a password, that would be avoided since they are easy to guess. So far, four techniques have been proposed to eliminate easy to guess passwords:

- *User education*. Users can be supported with guidelines for selecting good passwords. This approach gives poor results since users could avoid to follow these hints.
- *Computer-generated passwords*. A random password is assigned to each user. The drawback is that such a password usually is difficult to remember and

¹ Anderson has done a categorization of the intruders based on the behaviour once logged in. For details, see [1].

the user could write it down on a paper and the consequences can be unpredictable.

- *Reactive password checking.* The system administrator uses cracker programs to find guessable passwords. Of course, these checks are resources consuming and do not avoid the possibility that between two different checks some users could utilize easy to guess passwords. An attack in this period could be successful.
- *Proactive password checking.* A proactive password checker is a program that interacts with the user when he tries to change his own password. The proposed password is checked and the change is allowed only if it is hard to guess. This approach is the more promising and is the one that we are going to pursue in the next sections.

To simplify our discussion, we say that a password is a *weak* one if it is easy to guess; otherwise we say that it is a *strong* one.

A proactive password checker conceptually is a simple program. It holds a list of weak passwords that must be rejected. When the user wants to change his password, it checks for membership in the list. If the password is found the substitution is not enabled and a short justification is given; otherwise, the substitution is allowed.

However, a straightforward implementation of such a program is not suitable for two reasons: The list of weak words can be very long and, hence, must be stored in the secondary memory. Then, the time to check membership can be high. Since the check works in an interactive fashion, a long wait of the user cannot be accepted.

Various proactive password checkers that aim to reduce the *time* and *space* requirements of the trivial approach have been proposed, see for example [14, 10, 19, 7].

All these models are an improvement against the straightforward scheme. However, as showed in [2], they have a *low predictive power* when tested on new dictionaries. Indeed, a desirable feature of a proactive password checker is the ability to correctly classify passwords which do not belong to the initial set. We want a low *false negative* error rate, i.e., a small number of weak passwords classified as strong ones, and a low *false positive* error rate, i.e., a small number of strong passwords classified as weak ones.

To this aim, an interesting approach in designing a proactive password checker seems to be the one applied in [2]. Here, the problem of password classification is viewed as a *Machine Learning Problem*. The system, in a training phase, using dictionaries of examples of weak and strong passwords, *acquires the knowledge* to distinguish weak passwords from strong ones. This knowledge is represented by a *decision tree*. During the checking phase, the tree is used for classification. The experimental results reported in [2] show a meaningful enhancement on the error rates with respect to the previous solutions.

Since the approach seems promising, we have followed the same direction. We have tried to increase the power of such a checker exploring another key idea of machine learning in the construction of the decision tree: *The minimum*

description length principle. This principle essentially establishes that the best theory to describe a data set is the one that minimizes the sum of the length of the theory and of the length of the coded data using the theory. We have realized Hyppocrates, a prototype implementation and have done several tests to measure the performances of this new checker. The results confirm the validity of this choice.

We have chosen the name HYPPOCRATES for two reasons: First because it recalls the famous doctor of the antiquity². Then, because the name HYPPOCRATES appears into the logo of our University to remember the *Schola Medica Salernitana*, one of the first European “Academic” Institutions, supposed to be among the ancestors of the modern Universities.

Organization of the paper. In Section 2 we introduce the concept of classification with decision tree, describe pruning techniques which can increase the predictive power in classification tasks, and give a short description of the proactive password checker realized in [2]. In Section 3 we describe the principles on which we have realized HYPPOCRATES, our proactive password checker. In Section 4 we report experimental results and comparisons with previous solutions.

2 Classification with Decision Trees

In this section we describe the decision tree technique for classification used in the setting of proactive password checking in [2]. Moreover, we briefly describe the pessimistic pruning technique for increasing the predictive power of decision trees.

2.1 Decision Trees

A decision tree can be thought of as a mean to make decision. The root of the tree represents the starting point of the decision process while each leaf corresponds to a decision instance. To each internal node of the tree is associated an *attribute* that can be evaluated on an element that is undergoing the classification process, and which can assume many *values*. The values of the attribute are associated with the arcs outcoming from each node. A *decision* on a given element consists in a *path* from the root to a leaf determined by the values assumed by the attributes associated to each node along the path. More precisely, if the attribute associated to the current node can assume k different values, and the element we are considering assumes the i -th one, then the decision process will continue from the i -th child of the current node.

Since the problem we want to solve is *classification of passwords*, our aim is to realize a decision tree which can be used to establish if a password is a weak or a strong one. Hence, each leaf of the tree must correspond to a weak class of

² Actually, our system does not look after the ills of a bad password choice, it tries to do prevention avoiding bad choices and giving helpful hints to the users for good ones.

words or to a strong one. The following example reported in Figure 1 clarifies our discussion.

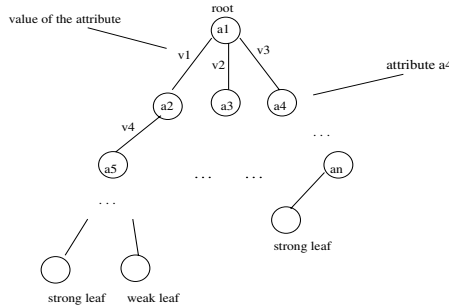


Fig. 1. Example of a decision tree for password checking

There are several methods to construct a decision tree. One of the most interesting is to consider the construction as a Machine Learning Problem. With this approach, we construct the tree by means of a *training* process in which a *set of examples* of weak and strong passwords is used to set up the tree. It is necessary to establish the *set of attributes* that must be evaluated in each node of the tree and a *criterion* to associate the attributes to the nodes. These are critical steps in the training phase. Indeed, the choice of examples, the choice of the set of attributes, and the criterion used for attribute selection at each node determine the performance of the tree (i.e., its size and its predictive power) when applied in classification tasks.

The learning process of a decision tree can be described as follows: Initially the root contains all the examples of the training set. Then, the training procedure tries to select an attribute for the root to partition the set of examples in subsets, according to an attribute selection strategy that optimizes a certain complexity measure. When this attribute is found then the partition is done and each “new” set of example is associate to a root’s child. The procedure is applied recursively on the children. The partition process on a node halts when it contains only elements of a given class or when there are no attributes to use for subsequent partitions. This means that, when the training process is complete, some leaves can store elements of different classes. The class of such a leaf is determined as the class of the greatest number of elements of the same type. Usually, in classification tasks which require an exact classification of the training set, this problem is bypassed using a file of exceptions in which can be stored all the misclassified elements.

2.2 Pruning Techniques

As we have seen, the decision tree can be used to establish if a password chosen by the user is weak or strong. If the password we are going to check belongs to

the training set, then its classification is surely correct³. However, if the password does not belong to the training set, its decision process proceeds along a path defined by the values assumed by the attributes on the password in each node. The hope is that the tree has stored *meaningful features* of the training set so that it is able to correctly classify a new password which shows characteristics similar to the examples of the training set.

In other words, our hope is that the decision tree has a good predictive power, too. However, notice that the decision tree can be *strongly dependent* from the training set. This means that we have a refined classification of the examples but a poor performance on unseen elements. To avoid this behaviour, we can simplify the structure of the tree using a *pruning technique*. The idea is that if we cut a subtree and replace it with a leaf, our tree can show a small error on elements of the training set, (i.e., misclassification) but it can gain predictive power on unseen dictionaries of words.

Many different techniques are known in literature to prune a decision tree. Among these, three are frequently used: *Cost-Complexity Pruning* [9], *Reduced Error Pruning* [16], and *Pessimistic Pruning* [16].

We briefly describe the Pessimistic Pruning Technique, since it has been used in designing HYPPOCRATES. We have chosen this technique since it allows to strongly reduce the size of the decision tree while maintaining a low error rate.

Pessimistic Pruning. To understand the logic of this technique notice that, during the training phase, an internal node of the tree contains well classified objects and misclassified objects. For example, suppose that in a node with K objects, J are misclassified and the remaining $K - J$ are well classified. The objects of this node are therefore partitioned in a subsequent step of the procedure. In the pruning phase, starting from a decision tree, we have to substitute a subtree with one leaf. We could decide to cut a subtree if the ratio $\frac{J}{K}$ is below a certain threshold. However, this is not a good strategy since this ratio does not give information about the *overall structure* of the tree. Therefore, we need a more complex approach.

Let S be a subtree of T which contains $L(S)$ leaves and let $E = \sum J$ and $N = \sum K$ be the number of misclassified objects and the number of objects of the leaves, respectively. The pessimistic pruning approach supposes that S will produce a bad classification of $E + \frac{L(S)}{2}$ elements in a set of N ones that do not belong to the training set. Denote with E' the number of misclassifications obtained if S is replaced with one of its best leaves. The pessimistic pruning technique replaces S when $E' + \frac{1}{2}$ is less than $stderr(E + \frac{L(S)}{2})$, where

$$stderr(E) = \sqrt{\frac{E(N - E)}{N}}$$

and is called the *standard error*.

³ Assuming that, if the password is misclassified by the decision tree, then it belongs to the file of exceptions.

To clarify the above procedure, suppose that we have two classes, and assume that S is a subtree with the following characteristics: $N = 4000$, $L(S) = 6$, and $E = 0$. Moreover, the objects which belong to the first class are 3998 while there are only two objects in the second class. This means that there is at least a leaf that classifies the elements of the second class and the other leaves that classify the 3998 elements of the first one. Now, the guessed error on N cases, not present in the training set, is $E + \frac{L(S)}{2} = 3.0$ with $stderr(E + \frac{L(S)}{2}) = \sqrt{\frac{3 \cdot 3997}{4000}} = 1.73$. If S is replaced with a leaf of the first class the error on the training set is equal to two and, hence,

$$E' + 1/2 = 2 + 1/2 = 2.5 < \left(E + \frac{L(S)}{2}\right) + stderr\left(E + \frac{L(S)}{2}\right) = 3.0 + 1.73$$

Therefore, according to the above rule, the pruning of S can be done.

2.3 The First Example of Proactive Password Checking with a Decision Tree

The first authors to suggest the use of decision trees in the design of a proactive password checker were Bergadano et al. in [2]. The decision tree stores information about the weakness or the hardness of the passwords used as examples in the training process. It can be viewed as a *high compress representation* of the dictionary of examples, where compression is obtained since prefixes common to many words need be stored only once.

The criterion chosen by Bergadano et al. in [2] to set up the decision tree was to maximize an information-theoretic measure called *the gain*. The idea is the following. Suppose that we have a set of attributes A . We have to select an element $a \in A$ for each node of the tree. Suppose that the training set is composed of p strong examples and n weak examples, and let $t = p + n$. When the selection procedure starts, all the elements of the training set are associated to the root of the tree. The *information* represented by the partition of the t examples is defined as:⁴

$$I(p, n) = -(p/t)\text{Log}(p/t) - (n/t)\text{Log}(n/t).$$

Although, if a is the selected attribute, we define

$$I(a) = \sum_{i=1}^s (t_i/t) I(p_i, n_i),$$

where s are the possible values that a can assume, and p_i (resp., n_i) out of the p (resp., n) are strong (resp., weak) examples having the i -th value for attribute a and, $t_i = p_i + n_i$. The *gain* of attribute a is then defined as $I(p, n) - I(a)$. Therefore, for each node, this procedure partitions the examples according to the attribute that maximize the gain.

⁴ In this paper all the logarithms are to the base 2.

Notice that Bergadano et al. [2], in their checker, use a well know system for learning decision tree called *C4.5* [17] which implements the above criteria. This system optimizes the constructed tree, to enhance predictive power on data sets different from the training set, with a pruning operation. This optimization is realized with the Pessimistic Pruning Technique described in the previous paragraph.

Once that the decision tree has been constructed, the checker is simply implemented. Indeed, it is a program that, when invoked, does a set of preliminary checks on some basic features of the password (e.g. length,...) and, then, uses the decision tree to decide class-membership.

3 HYPOCRATES: The Decision Tree

In the previous section we have described how to use decision tree techniques for password classification. We have pointed out that it is necessary to choose a criteria for attribute selection to construct a decision tree. Moreover, since the constructed decision tree strictly depends on the training dictionaries, we have also described some pruning techniques “to simplify” its structure to enhance predictive power. In this section we describe the choice we have done in the design of our system. First of all, we recall the minimum description length principle.

3.1 Minimum Description Length Principle

The Minimum Description Length Principle has been introduced by Rissanen [18]. Conceptually, it says that the best theory to describe a data set is the one that minimizes the sum of two elements:

- the size of the theory
- the length of the coded data using the theory

In our setting this principle essentially means that the shorter tree that can be constructed to correctly classify the training set is the best.

An intuitive justification of this principle, applied to the decision trees environment, can be achieved if we consider the problem of constructing a decision tree as a *communication problem* in which we want to minimize the communication cost.

To this aim, we consider a scenario in which there are two users, *A* and *B*. They share a table with several columns. In the first column there are the words (weak and strong). The subsequent columns are indexed by some attributes of the words and are filled in with the corresponding values of the attribute for each word. The last column contains the classification of the words and it is stored *only* by *A*. The problem that *A* has to solve is *to send* this last column to *B* with the *minimum number* of bit.

Suppose that this column has the following form:

$$W, W, S, S, W, S, W, S, S, S, S, S, S, W$$

Table 1. Table stored by A and partially shared with B

words	attr ₁	attr _n	class
password	val ₁	val _j	W
proactive	val ₃	val ₁	W
...
...
x1[!T?	S
checker	val _s	val _r	W

where W corresponds to weak and can be encoded with a bit equal to 0 and S corresponds to strong and can be coded with a bit equal to 1. If we send directly this encoding of the column, then we need to transmit 14 bits. In general, we can apply the following transmission strategy: Suppose that the sequence we have to send has length n . If the sequence has k bits equal to 1, we can send the value k and an index that establishes which of the possible $\binom{n}{k}$ k weight sequence corresponds to the given sequence. The total number of required bits is quantified by the function

$$L(n, k, b) = \text{Log}(b + 1) + \text{Log}\left(\binom{n}{k}\right)$$

In this formula b is a fixed constant greater than k , enough big to ensure that $\text{Log}(b + 1)$ bits can represent the value of k .

In the above example, with $b = 9$, since $n = 14$, and $k = 9$ it holds that $L(14, 9, 9) = 14.289$. Hence, in this case this coding technique is not cheaper than the trivial transmission. However, we can do better.

Notice that we can enhance in our transmission problem using the *common table*. For example, suppose that with respect to a *given attribute*, which takes two values, the classified words can be split into two sets: W, W, S, S, W, S, W , and S, W, S, S, S, S, S . If we apply the previous strategy to the two sequences, then we have that the total transmission cost is equal to

$$L(7, 3, 6) + L(7, 6, 6) = 13.551$$

which is an improvement compared to the trivial transmission approach.

The key point in this example is that if we partition the starting set according to a given attribute, the overall transmission complexity can be reduced.

The function $L(n, k, b)$ can be approximated applying the Stirling function. More precisely, we have that

$$L(n, k, b) = nH\left(\frac{n}{k}\right) + \frac{4\text{Log}(n)}{2} - \frac{\text{Log}(k)}{2} - \frac{\text{Log}(n-k)}{2} - \frac{\text{Log}(2\pi)}{2} - \text{Log}(b) + O(1/n),$$

where $H(p) = -p\text{Log}(p) - (1-p)\text{Log}(1-p)$ is the binary entropy function.

Coming back to decision trees, our attribute selection procedure in the training phase works as follows: Starting from the root, which at the beginning holds all the examples, the procedure tries to select the attribute inducing a partition on the data set which minimizes the function $L()$, our complexity measure. The

procedure is then applied recursively on the children of the root, and so on. It halts when there is no need of further partitions since the objects of the node belong to the same class or when all the attributes have been selected.

We have chosen the minimum description length principle in the design of our system because it seems that this measure is more refined than the gain and because it considers the global cost of the representation of the tree. This cost is ignored by the gain method. Experimental results corroborate these intuitions.

3.2 Attributes Used for Classification

As stated in the previous section, we need to define the attributes that will be used during the training and testing phases.

We report below the attributes we have used in our system. Some of them, namely C, D, G , are similar to the attributes used in [2]. We notice that some attributes presented below, namely A, B, C, D are evaluated on a single character of the password that is undergoing the classification, the attribute E considers pair of characters and the last two attributes, i.e., F, G , are evaluated on the whole password. We have defined the following attributes.

- A : Returns a value in $\{0,1,2\}$ when the character belongs to one of the following sets: $\{\text{vowels} + \text{consonants}\}, \{\text{digits}\}, \{\text{other}\}$.
- B : Returns a value in $\{0,1,2,3\}$ when the character belongs to one of the following sets: $\{\text{vowels}\}, \{\text{consonants}\}, \{\text{digits}\}, \{\text{other}\}$.
- C : Returns a value in $\{0,1,2,3\}$ when the character belongs to one of the following sets: $\{\text{vowels}\}, \{\text{n,m,r,l,c,g,k,x,j,q,h}\}, \{\text{t,d,b,p,f,v,y,w,s,z}\}, \{\text{digits}\}, \{\text{other}\}$.
- D : Returns a value in $\{0,1,2,3,4,5,6\}$ when the character belongs to one of the following sets: $\{\text{vowels}\}, \{\text{n,m,r,l}\}, \{\text{c,g,k,x,j,q,h}\}, \{\text{t,d,b,p}\}, \{\text{f,v,y,w,s,z}\}, \{\text{digits}\}, \{\text{other}\}$.
- E : This attribute considers pair of characters. Returns a value in $\{0,1,2,3,4,5,6\}$ when the character belongs to one of the following sets (here we use the notation $v=\text{vowel}$, $c=\text{consonant}$, $d=\text{digit}$, $o=\text{other}$, $n=\text{null}$): $\{vn, cn, dn, on\}, \{vv, vc, cv, cc\}, \{vd, dv\}, \{vo, ov\}, \{cd, dc\}, \{co, oc\}, \{dd, do, od, oo\}$.
- F_i (with $i = 6, 7, 8$): Returns 0 if the words is shorter then i characters, 1 otherwise.
- G_i (with $i = 1, \dots, 8$): Returns 1 if the words contains at least i non-alphanumeric characters, 0 otherwise.

At this point we notice that we need to solve a first problem, that is to decide which classification criterion must be applied when testing an attribute on a position p on a password whose length is less than p .

For the first problem we have identified two possible strategies: Classify the character as a vowel, i.e. as belonging to the class with lowest “power”, or classify it as belonging to a special class. The experiments have shown that the first solution is more suitable since we obtain higher compression ratio and lower error rates.

4 Tests and Comparisons

In this section we report the results of the experiments we have run to test the performances of our system. We start by describing the results obtained using the first eight characters of each word in the dictionary to classify the passwords. We will also report the experiments we have done using dynamic attributes, i.e., allowing the system to use a number of characters different from eight for the classification process.

4.1 Testing the Predictive Power

The main problem with the proactive password checkers is that the users can choose passwords that are contained in some dictionary that has not been used during the training phase. This means that the decision tree must learn, in a certain sense, the characteristic of the dictionaries used for the training phase more than the dictionaries themselves. Only in this way, it can be actually used for determining whether a password is good or bad.

To check the predictive power achieved by the trees constructed by HYPPOCRATES, we have run several training phases, followed by testing operations on dictionaries not used for constructing the trees. As we are interested in testing the predicting power of the tree, we do not use the dictionaries of exceptions generated by the training phase. Notice, however, that the check on the exception dictionaries will simply decrease the errors in case there are misclassified words that are common to the weak dictionaries used for the training and weak dictionaries in the testing set.

4.2 The Dictionaries

As we have seen in the previous sections, the problem of password classification is seen as a machine learning problem. Before presenting the experiments, let us describe the dictionaries we have used for the training and the testing phases.

We have used for the training set a big dictionary of weak passwords, we call **Total.clean**, consisting of 1,668,118 *different* words taken from various languages, e.g., English, Italian, Spanish, French, and some thematic dictionaries, e.g., computer, science, sport, cinema. The size of this weak dictionary approaches to 17 Mbytes.

We use two different types of strong dictionaries. The first type, we simply call **Strong**, is composed by words that have been randomly generated using the characters reported in Table 2 of Appendix. The second one, we call **Very.Strong**, is composed by randomly generated words, with at least two *strong characters* (see Table 2 for the list of strong characters). We discuss the size of these dictionaries in the next sections.

We have also used noise dictionaries generated by inserting or substituting one randomly chosen letter in the weak password with a strong character.

For space limits, we only report the results of the test ran over four weak testing dictionaries, we call **Dic.1**, **Dic.2**, **Dic.3**, **Dic.4**. These dictionaries

Table 2. The Charset

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
O	P	Q	R	S	T	U	V	W	X	Y	Z	a	b	c
d	e	f	g	h	i	j	k	l	m	n	o	p	q	r
s	t	u	v	w	x	y	z	0	1	2	3	4	5	6
7	8	9	Strong Characters											
!	"	#	\$	%	&	()	*	+	,	-	.	/	:
=	?	{	}]	^	['	'	@	;	<	>		~
\	%													

contain almost 30,000 different words each, taken from four books. From these dictionaries we have constructed noisy testing dictionaries `Dic.i.1`, `Dic.i.2`, `Dic.i.3`, `Dic.i.4` and `Dic.s.1`, `Dic.s.2`, `Dic.s.3`, `Dic.s.4` by inserting, respectively substituting, one character with a strong one.

The strong testing dictionaries, `Dic.Strong`, `Dic.Vstrong` have been constructed using the same criterion described above.

4.3 Fixed Length Attributes

In this subsection we describe the experiments done using the first eight characters of the password for the classification process. The choice of this length is due to the fact that Unix operating systems allow passwords at most eight characters long.

Optimizing the Compression Ratio. The first experiment we have run is directed to study the dependence of the compression ratios and error rates with respect to the size of the strong dictionary.

As we have seen in the previous sections, we need a dictionary of weak passwords and a dictionary of good ones, and the size of the training set is the sum of the size of both these dictionaries.

It is likely to think that the smaller is the training set, the smaller is the tree (trained and pruned) that classifies the training set. On the other hand, it is likely to think that the smaller is the training set, the higher is the error occurring when the tree is used to classify passwords that do not belong to the training set.

Recall that from the point of view of password security, it is important that the checker does not accept weak passwords, while rejecting a strong password simply translates into a request for the user to type a new password in.

Thus since we want to reduce the training set, we cannot reduce the size of the weak dictionary, as all these passwords will be correctly classified as weak by the checker, but we should simply reduce the size of the strong dictionary.

Based on this idea, we have run several experiments using `Total.clean` as weak dictionary and a randomly generated strong dictionary whose size ranges between the 5% and the 100% of the size of the weak dictionary. The results of these experiments confirm that if the size of the strong dictionary increases then also the size of the decision tree increases. We have used as strong training dictionaries both **Strong** and **Very.Strong**. All the experimental results show that using the second dictionary leads to higher compression ratios and lower error rates.

In Table 3 we report the compression ratios obtained using this approach to construct the training set for the experiments described in Section 4.3. In this table, the coefficients $C1$ (resp., $PC1$), are computed as the ratio between the size of the trained (resp., pruned) tree and the size of the weak dictionary, while the coefficients $C2$, (resp., $PC2$) are computed as the ratio between the size of the trained (resp., pruned) tree and the sum of the sizes of the weak dictionary and the dictionary of exceptions. This means that the lower is the value of the coefficient, the higher is the compression ratio achieved by the experiment.

Table 3. Compression Ratios

Exp. no.	Strong dic. %	C1	PC1	C2	PC2
3.1	5	1.060616%	0.030250%	1.066092%	0.248871%
3.2	10	1.500487%	0.033102%	1.517905%	0.571697%
3.3	20	2.094764%	0.049583%	2.138685%	1.077475%
3.4	30	2.520016%	0.053936%	2.590119%	1.364063%
3.5	40	2.871475%	0.059337%	2.963804%	1.781389%
3.6	50	3.166717%	0.057475%	3.282923%	2.212565%
3.7	60	3.413457%	0.058452%	3.549247%	2.417352%
3.8	70	3.642529%	0.058918%	3.795033%	2.745247%
3.9	80	3.820715%	0.062247%	3.995641%	2.975094%
3.10	90	4.020887%	0.064423%	4.209734%	3.227532%
3.11	100	4.206417%	0.063667%	4.410313%	3.541192%

Surprisingly, the error rate obtained in these experiments show that if the size of the strong dictionary decreases, the error obtained on the weak and noise testing dictionaries decreases too. This means that we can obtain lower error by using smaller trees. On the other hand, the error rate decreases for the strong testing dictionaries if the size of the strong dictionary used for the training increases. However, while the error rate on the weak dictionaries is acceptable, the error on the noise and strong dictionaries often greater than 10%, that is, of course, unacceptable.

Decreasing the Error on Noise Dictionaries. One of the basic problem with the training strategy presented in Section 4.3 is that the error rate ob-

tained on the noise dictionaries is too high. In order to decrease this error, we have run some experiments in which we substitute the weak dictionary in the training phase, with a noisy one. More precisely, we construct two dictionaries `Total.clean.ins` and `Total.clean.sub`, obtained from `Total.clean` by inserting, resp., substituting, a strong character in a random position within each password of the dictionary. The results of these experiments show that the error rate obtained on noise dictionaries is close to zero.

However these kind of trainings have three side effects: First, the size of the trees obtained is greater than before. The reason for this is that the “distance” between the strong and the weak dictionaries is smaller and, thus, more attributes are needed to correctly classify the training set. Second, the error rate on the weak and strong dictionaries is too high. Third, if the tree has been constructed using a noise dictionary obtained by inserting (resp., substituting) a strong character, this tree has lower error on testing dictionaries constructed using the same type of noise, i.e., insertion (resp., substitution), but has higher error on testing dictionaries obtained by substituting (resp., inserting) strong characters.

Decreasing Error by Mixing. The main problems with the experiments presented in this section are the high error rate on strong and weak dictionaries and the dependence of the error on different noise dictionaries. To solve these problems, we have constructed a new dictionary `Total.clean.all` obtained from `Total.clean` as follows: Each word in the original dictionary has the same probability to be put in the new dictionary unchanged, with a strong character substituted, with a string character inserted in a random position.

As we will see in the next section, this strategy leads to good compression ratios and low error rates.

Table 4. Comparisons with Other Password Checkers

Checker	training dic.(byte)	Data (byte)	C1 (PC1)*	Exception (byte)	C2 (PC2)*	time (sec)
3.2 Trained	17183224	257832	1.50049%	2993	1.51790%	5774
3.2 Pruned	17183224	5688	0.03310%*	92548	0.57170%*	5825
3.6 Trained	17183224	544144	3.16672%	19968	3.28292%	8878
3.6 Pruned	17183224	9876	0.05747%*	370314	2.21256%*	8949
ProCheck	1305234	32121	2.46094%	6548	2.96261%	?
Crack	16626766	7867297	47.31706%	0	47.31706%	121

4.4 Comparing Hypocrates with Other Checkers

In this subsection we compare the performances of Hypocrates, using the results obtained by training the decision tree using the dictionary described in Section 4.3, with respect to two password checkers, Crack and ProCheck presented, respectively in [13] and [2]. We have used `Total.clean` to “train” Crack, as it leads to lower error rate, and we have used the best compression ratios reported in [2]. In Table 4 we report the compression ratios for each password checker. In the last column of this table we report the time required for the training phase. We notice this column is incomplete because the time required for the training of ProCheck are not reported in the paper.

In Tables 6, 7, 8, 5 we report the error rates obtained by classifying the testing dictionaries described in Section 4.2. From the results it is immediate that Hypocrates achieves better compression ratios and lower error rates with respect to the other checkers. Recall that the authors in [2], compared ProCheck with many other password checkers, showing that their system was significantly better than the other considered checkers.

Table 5. Testing Strong Dictionaries

Checker	Error (%)	
	Dic.Strong(5%*)	Dic.VStrong (5%*)
3.2 Trained	13.8097 %	7.5911 %
3.2 Pruned	25.4671 %	15.5040 %
3.6 Trained	13.5496 %	3.1139 %
3.6 Pruned	14.2640 %	4.4056 %
ProCheck	21.7712 % *	20.1322 % *
Crack	13.6357 %	13.6671 %

Table 6. Testing Weak Dictionaries

Checker	Error (%)			
	Dic.1	Dic.2	Dic.3	Dic.4
3.2 Trained	0.0586%	0.0056%	0.1759%	0.3387%
3.2 Pruned	0.0586%	0.0000%	0.1971%	0.2909%
3.6 Trained	0.1298%	0.0641%	0.4367%	0.5254%
3.6 Pruned	0.1340%	0.0362%	0.3427%	0.5210%
ProCheck	1.5410%	0.9250%	2.5202%	1.4805%
Crack	15.0042%	19.5954%	17.1898%	17.2325%

Table 7. Testing Noise Dictionaries (Insertion)

Checker	Error (%)			
	Dic.i.1	Dic.i.2	Dic.i.3	Dic.i.4
3.2 Trained	1.3735%	1.3541%	2.0107%	2.5269%
3.2 Pruned	1.3735%	0.9362%	1.2677%	2.1058%
3.6 Trained	2.7136%	2.5800%	3.5817%	4.4547%
3.6 Pruned	3.0193%	2.8642%	3.7000%	5.0191%
ProCheck	2.1315%	1.2427%	2.9084%	2.2925%
Crack	40.5193%	47.1511%	44.9004%	43.5698%

Table 8. Testing Noise Dictionaries (Substitution)

Checker	Error (%)			
	Dic.s.1	Dic.s.2	Dic.s.3	Dic.s.4
3.2 Trained	1.5704%	1.3736%	2.3170%	2.5052%
3.2 Trained	1.1307%	1.0978%	1.5952%	2.1883%
3.6 Trained	3.1826%	2.8085%	3.9002%	4.4764%
3.6 Pruned	3.1868%	2.7862%	3.7455%	4.7239%
ProCheck	3.7898%	2.6859%	4.8282%	3.7774%
Crack	47.6424%	51.1047%	48.2577%	47.7727%

4.5 Dynamic Length Attributes

In the previous subsections we have described the results obtained by Hypocrates when classifies the passwords using its first eight characters. However, many systems allows passwords with more than eight characters. Our system allows the possibility to select the maximum number of characters of the password to be used for the classification process.

Choosing the Optimal Length. As we now have the possibility to choose an arbitrary number of characters for the classification process, we need to decide which is the optimal one for a given training set.

Intuitively, this length should depend on the average length of the passwords in the training set. Indeed, if we use few characters, the decision tree should not have enough information to correctly classify the training set. On the other hand, it should be useless to check the i -th character in the password if i is much greater than the average length, since few passwords will be classified by means of that character.

To argument this idea, we have run several experiments whose description follows: The training set consists of the dictionary `Total.clean.all` described in Section 4.2, whose average password length is 9.3. We have constructed strong dictionaries, with size ranging from 10% to 100% of the size of the weak dictio-

nary, with the same average password length. We allow the system to use the first 6, 8, 12, 16 characters of the password for the classification.

The results of these experiments show that, if we look at experiments that use the same number of characters for the classification, as the size of the strong dictionary increases, the size of the decision tree increases too. Moreover, the error rate on weak and noise dictionary decreases, while, as before, the error rate on the strong dictionary decreases.

It is interesting to see that the lowest error rates are achieved when the number of characters used for classification approaches to the average length of the passwords in the dictionary.

5 Conclusions

In this paper we have described a new realization of a proactive password checker. We have explored the possibility of using the Minimum Description Length Principle, a key idea of Machine Learning, in the context of proactive password checker. The results reported in the tests and comparisons section of our work indicate that the pursued approach is suitable. HYPOCRATES has been tested on a 450 MHz Pentium III Linux box with 8Gb EIDE hard disk. The code of the prototype implementation has been written using the ANSI C language and the graphical interface has been realized for X-Windows system. We are working on realizing patches for the unix-like passwd commands. A beta version of the software will be available from our Web site (<http://www.security.unisa.it>).

References

1. J. Anderson. Computer Security Threat Monitoring and Surveillance. Fort Washington, PA: James P. Anderson Co. April 1980.
2. F. Bergadano, B. Crispo and G. Ruffo. High Dictionary Compression for Proactive Password Checking. ACM Transactions on Information and System Security. Vol. 1, No. 1, November 1998, Pages 3-25.
3. M. Bishop. Anatomy of a Proactive Password Checker. Proceedings of the Third UNIX Security Symposium, pp. 130-139, September 1992.
4. M. Bishop. Proactive Password Checking. Proceedings of the Fourth Workshop on Computer Security Incident Handling, pp. W11:1-9 (Aug. 1992).
5. M. Bishop. Improving System Security via Proactive Password Checking. Computers and Security, 14(3) pp. 233-249 (1995)
6. M. Bishop. Password Management. Proceedings of COMPCON 1991, pp. 167-169 (Feb. 1991).
7. B. Bloom, Space/Time Trade-offs in Hash Coding with Allowable Errors. Communications of ACM, July 1970.
8. V. Boyko, P. MacKenzie, S. Patel. Provably Secure Password-Authenticated Key Exchange Using Diffie-Hellman. Proceedings of Eurocrypt 2000, LNCS vol. 1807, pp. 156-171.
9. Breiman, L., J. H. Friedman, R. A. Olshen, and C. J. Stone. Classification and Regression Trees. Wadsworth & Brooks/Cole, 1984 Advanced Books & Software. Pacific Grove, CA.

10. C. Davies and R. Ganesan. Bapasswd: A new proactive password checker. Proceedings of the 16th National Conference on Computer Security. Baltimore, MD, Sept. 20-23.
11. J. Katz, R. Ostrovsky, M. Yung, Efficient Password-Authenticated Key Exchange Using Human-Memorable Passwords. Proceedings of Eurocrypt 2001 LNCS vol. 2045, pp. 475-495.
12. D. Klein. Foiling the Cracker: A Survey of, and Improvements to, Password Security. Proceedings of the Fifth Data Communications Symposium. September 1977.
13. A. Muffett. Crack 5.0. USENET News.
14. J. B. Nagle. An Obvious Password Detector. USENET News 16, 60.
15. B. C. Neuman and T. Tso. Kerberos: an authentication service for computer networks. IEEE Trans. Commun., 32. 33-38, 1994.
16. J. R. Quinlan. Simplifying decision Trees, Int. J. of Man Machine Studies, 27 Academic Press Limited, London, 1987
17. J. R. Quinlan. C4.5: Program for Machine Learning, Morgan Kaufmann Publishers Inc., San Francisco, CA
18. J. Rissanen, Stochastic complexity and modeling, Ann. Stat. 14. 3, 1080-1100.
19. E. Spafford. OPUS: Preventing Weak Password Choices. Computers and Security, No. 3, 1992
20. R. Stalling. Network and Internetwork Security. Prentice Hall, Englewood Cliffs, New Jersey

Lenient/Strict Batch Verification in Several Groups

Fumitaka Hoshino, Masayuki Abe, and Tetsutaro Kobayashi

NTT Information Sharing Platform Laboratories, NTT Corporation
1-1 Hikari-no-oka, Yokosuka-shi, Kanagawa-ken, 239-0847 Japan
`{fhoshino,abe,kotetsu}@isl.ntt.co.jp`

Abstract. Batch verification is a useful tool in verifying a large number of cryptographic items all at one time. It is especially effective in verifying predicates based on modular exponentiation. In some cases, however, the items can be incorrect although they pass batch verification together. Such leniency can be eliminated by checking the domain of each item in advance. With this in mind, we investigate if the strict batch verification can remain more effective than separate verification. In this paper, we estimate the efficiency of such strict batch verification in several types of groups, a prime subgroup of \mathbb{Z}_p with special/random prime p and prime subgroups defined on elliptic curves over \mathbb{F}_p , \mathbb{F}_{2^m} and \mathbb{F}_{p^m} , which are often used in DL-based cryptographic primitives. Our analysis concludes that the efficiency differs greatly depending on the choice of the group and parameters determined by the verifying predicate. Furthermore, we even show that there are some cases where batch verification, regardless of strictness, loses its computational advantage.

1 Introduction

In cryptographic protocols verification of items such as signatures, zero-knowledge proofs, and ciphertexts plays an important role to maintain robustness. In large-scale applications, such verification could dominate efficiency since these items are typically verified by a central server while generated by each player. A typical example would be a bank serving in an off-line e-cash system, which verifies all electronic coins, i.e., the bank's signatures, spent each day [1]. Another good example would be electronic voting schemes where millions of voters cast encrypted ballots and a group of centralized servers shuffles and opens them in a verifiable way [2,3,4,5]. When real-time response is not required, batch verifying items would reduce the cost. In [2,6,7,8], it was shown that a type of verification predicates based on modular exponentiation can be efficiently batch verified.

It was shown in [9], however, that a set of items that is surely rejected in individual verification can be accepted in the batch mode with high probability depending on the group that defines the items. Intuitively, it happens when some elements of the items, which are supposed to be in a group, are in fact out of the group in such a way that the deviating parts eliminate each other in combination

so that they are accepted in batch mode. As observed in [9], such erroneous items might not be a critical problem when the items are signatures since it is only the legitimate signer who can generate the erroneous signatures. *However, when the verifying items are zero-knowledge proofs, it could be a serious threat since the mistakenly accepted items may be used in further processing.*

In this paper, we consider *strict* batch verification that provides virtually the same confidence level as that provided by separately verifying each item. We estimate the efficiency of such strict batch verification in several types of groups such as a prime subgroup of \mathbb{Z}_p with special/random prime p and prime subgroups defined on elliptic curves over \mathbb{F}_p , \mathbb{F}_{2^m} and \mathbb{F}_{p^m} , which are often used in DL-based cryptography. We show that the efficiency level differs greatly depending on the choice of the group and the parameter determined by the verifying predicate. Furthermore, we even show that there are some cases where batch verification, regardless of the strictness, loses its computational advantage. This result would be an independent interest as batch verification is supposed to be faster than separate verification.

The rest of the paper is organized as follows. In Section 2, we define the items to check, and review some verification methods including separate verification and lenient/strict batch verification. In Section 3, our results are summarized. Section 4 introduces a criterion for the advantage of batch verification. Based on this criterion we estimate the cost of batch verification in Section 5.

2 Preliminaries

2.1 Separate Verification of DL-Based Items

Let g be a generator of a cyclic group of order q . In this section, we use multiplicative notation for the group operation and all arithmetic operations are done in the group unless otherwise noted. Let \mathcal{I}_n^k be a collection of n items for verification. Let I_i^k be the i -th item in \mathcal{I}_n^k defined as

$$I_i^k = (e_{i1}, \dots, e_{ik}, h_{i0}, h_{i1}, \dots, h_{ik}) \in \mathbb{Z}_q^k \times \langle g \rangle^{k+1}$$

for some $k \geq 1$. Separate verification, $V_{\text{sep}}(\mathcal{I}_n^k)$, consists of two phases

$$\text{Domain test: } h_{ij} \stackrel{?}{\in} \langle g \rangle \quad \text{for } 1 \leq j \leq k, 1 \leq i \leq n, \text{ and} \quad (1)$$

$$\text{Separate relation test: } h_{i0} \stackrel{?}{=} \prod_{j=1}^k h_{ij}^{e_{ij}} \quad \text{for } 1 \leq i \leq n. \quad (2)$$

We denote $V_{\text{sep}}(\mathcal{I}_n^k) = \text{true}$ if \mathcal{I}_n^k passes the above tests. The items in \mathcal{I}_n^k are defined to be correct if $V_{\text{sep}}(\mathcal{I}_n^k) = \text{true}$. It is stressed that the domain test does not apply to h_{i0} because if an item satisfies the predicate h_{i0} must also pass the domain test.

2.2 Lenient/Strict Batch Verification

Without loss of generality, we assume that h_{ik_x}, \dots, h_{ik} are common to all i and the rest of the bases are different in each item. To simplify the notation, we omit suffix i if it's unnecessary, thus h_{ik_x}, \dots, h_{ik} are denoted as h_{k_x}, \dots, h_k . Lenient batch verification $V_{\text{bat}}(\mathcal{I}_n^k)$ consists of

$$\begin{aligned} \text{Lenient domain test:} & \quad (\text{the same as in } \textcolor{red}{\text{[1]}}) \\ \text{Batch relation test:} & \quad 1 \stackrel{?}{=} \left(\prod_{i=1}^n \prod_{j=0}^{k_x-1} h_{ij}^{r_i e_{ij}} \right) \left(\prod_{j=k_x}^k h_j^{\hat{e}_j} \right), \end{aligned} \quad (3)$$

where $e_{i0} = -1$ and $\hat{e}_j = \sum_{i=1}^n r_i e_{ij} \in \mathbb{Z}_q$ for $r_i \in_U R \subseteq \mathbb{Z}_q$. We denote $V_{\text{bat}}(\mathcal{I}_n^k) = \text{true}$ if \mathcal{I}_n^k passes the above tests. From the results described in [\[9\]](#), it is shown that if all items are taken from the correct domain and $V_{\text{bat}}(\mathcal{I}_n^k) = \text{true}$, then $V_{\text{sep}}(\mathcal{I}_n^k) = \text{true}$ with probability $1 - 1/|R|$.

In the above, note that h_{i0} is still outside the domain testing. We define *strict batch verification* $V_{\text{str}}(\mathcal{I}_n^k)$ as a combination of

$$\begin{aligned} \text{Strict domain test:} & \quad h_{ij} \in \langle g \rangle \quad \text{for } 0 \leq j \leq k, 1 \leq i \leq n, \text{ and} \\ \text{Batch relation test:} & \quad (\text{the same as in } \textcolor{red}{\text{[3]}}) \end{aligned} \quad (4)$$

Note that h_{i0} is now included in the domain test. We denote $V_{\text{str}}(\mathcal{I}_n^k) = \text{true}$ if \mathcal{I}_n^k passes these tests. When $V_{\text{str}}(\mathcal{I}_n^k) = \text{true}$, this implies that $V_{\text{sep}}(\mathcal{I}_n^k) = \text{true}$ with probability $1 - 1/|R|$.

To see the implications of the strict domain test, we review the results of [\[9\]](#), which state that batch verification without a strict domain test does not provide the same security as does separate verification. The example abuses the implicit assumption that h_{i0} is taken from the correct domain. Here, we assume that $\langle g \rangle$ is a multiplicative subgroup in \mathbb{Z}_p where $\text{ord}(g) = q$, $p = 2q + 1$, and q is prime. Let $\mathcal{I}_2^k = \{(e_{i1}, \dots, e_{ik}, h_{i0}, h_{i1}, \dots, h_{ik})\}_{i=1,2}$ be correct items with regard to the separate verification predicate. Note here that -1 is not in $\langle g \rangle$. Then, let $\tilde{\mathcal{I}}_2^k = \{(e_{i1}, \dots, e_{ik}, (-1)h_{i0}, h_{i1}, \dots, h_{ik})\}_{i=1,2}$. Note here that $\tilde{\mathcal{I}}_2^k$ fails in the separate verification, but it passes the batch verification with probability $1/2$. This happens when r_1 and r_2 are both even or odd.

2.3 Overlap Factor

In this section, we introduce the overlap factor, which is an essential parameter in the evaluation of verification cost in later sections.

The advantage of batch verification is based on the fact that

- computing $(n \times k)$ -base exponentiation is faster than computing k -base exponentiation n times, and
- exponents for overlapped bases can be wrapped up.

Accordingly, batch relation Predicate (3) can be computed faster than repeatedly computing separate Predicate (2) n times. We refer to Algorithm 14.88 in [10] for an exponentiation algorithm with multiple bases. When many of the bases are common to all the items, the advantage clearly grows. We define *overlap factor* k_o as $k - (k_x + 1)$ such that it represents the number of bases commonly included in all the items.

- $k_o = 0$ No base is common among the items. This occurs, for instance, in the predicate of zero-knowledge proofs that proves equality of the discrete logarithms, $\log_{g_1} y_1 = \log_{g_2} y_2, \dots$
- $k_o = 1$ Only one base is common among the items. Such a case would be observed in applications where all items are generated based on a single generator, say g .
- $k_o \geq 2$ More than two bases are common. Such a case would be observed when the items are based on the representation problem such as the Okamoto identification scheme [11] or Brand's electronic cash scheme [1].

The advantage of batch verification is maximized when $k_o = k - 1$ where all but one base are common and minimized when $k_o = 0$ where all bases are different.

3 Summary of Our Results

We analyze the advantage of batch verification by comparing it to the separate verification. The advantage is estimated w.r.t. the computational cost. We say that it has advantage if it saves more than one exponentiation per item. Our results are summarized in Table 1. See Section 5 for the classification of groups and specific schemes of strict batch verification. The symbols **a**, **d** and **D** in Table 1 respectively denote

- a**: Strict batch verification is unconditionally advantageous,
- d**: Strict batch verification is disadvantageous, and
- D**: Lenient batch verification is unconditionally disadvantageous.

The lower case symbols (**a**, **d**, **q**, and **e**) also have an additional meaning,

“Lenient batch verification is unconditionally advantageous,”

and the upper case symbols (**E** and **D**) represent

“Strict batch verification is unconditionally disadvantageous.”

What is interesting in Table 1 is

- I-1 Almost all lower case areas are occupied by symbol **a**.
- I-2 Symbols **d** and **D** have completely different reasons why the strict batch verification is disadvantageous.

Item I-1 means that strict batch verification can be performed efficiently on almost all practical groups on which lenient batch verification can be performed efficiently. Item I-2 implies the following.

- Symbol **d** implies that no effective domain test is found. If an effective test is found, strict batch verification becomes advantageous. The symbol **q** is a special case of **d**.
- Symbol **D** implies that batch verification cannot be faster than separate verification although an effective domain test is available. This means that the separate verification is sufficiently fast by itself. Symbols **E** and **e** are special cases of **D**.

Note that **D** never means that the binary field is unsuited to large-scale applications.

Table 1. Effectiveness of Batch Verification.

Section	Group	$k_o = 0$	$k_o = 1$	$k_o = 2$	$k_o \geq 3$
5.1	Subgroup of \mathbb{Z}_p^* (random prime)	d		q	a
5.2	Subgroup of \mathbb{Z}_p^* (special prime)				
5.3	Prime Field				
	OEF (random curve)				
5.4	Binary Field (random curve)				
5.5	OEF (special curve)	E	e		
	Binary Field (special curve)	D			

Parameter k_o is the number of common bases among all verification items.

a : Strict batch verification is unconditionally **a**dvantageous.

d : Strict batch verification is **d**isadvantageous if no effective test is found.

q : Strict batch verification is advantageous if the q -th power is fast,

e : Strict batch verification is advantageous if the extension degree is small.

E : Lenient batch verification is advantageous, if the **E**xtension degree is small.

D : Lenient batch verification is unconditionally **D**isadvantageous.

4 Basic Concepts for Analysis

4.1 Gain and Loss

In this section, we introduce the concept of gain and loss, which provides a simple criterion for the advantage of batch verification. In the rest of this paper, we estimate computation costs by the number of group operations in $\langle g \rangle$ and assume that all other arithmetic operations, equality tests, and random number generations are done for free.

The gain, c_{gain} , is defined as the difference in the cost per verification item between separate relation test **(2)** and batch relation test **(3)**. That is, let C_{sep} and

C_{bat} be the computational cost of Predicate (2) and Predicate (3) respectively, then

$$c_{\text{gain}} = (C_{\text{sep}} - C_{\text{bat}})/n \quad (5)$$

and the loss, c_{loss} , is defined as the cost to ensure $h_0 \in \langle g \rangle$ for strict batch verification. That is the difference in the cost per verification item between lenient domain test (II) and strict domain test (4).

The concept of gain and loss provides a simple criterion for the advantage of batch verification: If the loss is less than the gain, then the batch verification is meaningful, otherwise it does not make any sense. The advantage of the batch verification is given by

$$c_{\text{loss}} < c_{\text{gain}} \quad (6)$$

For discussion on lenient batch verification, set $c_{\text{loss}} = 0$ since the domain test is the same as that in separate verification, while for strict batch verification, c_{loss} must be estimated.

4.2 Generic Evaluation of the Gain

This section estimates c_{gain} more precisely without specifying an underlying group. For this purpose, we focus on an abstract exponentiation scheme called the ‘optimal’ exponentiation scheme, and we give an estimation of c_{gain} for such optimal exponentiation scheme. The concrete scheme may be the binary method, the signed binary method, or something window method [10]: however, this is not specified because it varies with the choice of the underlying group. We will discuss such variations in later sections in this paper.

Let G be the group we focus on where c_m is the computational cost of a multiplication on G and c_s is the cost of a square on G . Let N_m be the average number of multiplications used in the optimal exponentiation scheme and N_s be the average number of squares required by the scheme. The average cost of the exponentiation scheme on G is $N_m c_m + N_s c_s$. By extending the scheme [8, 10], k -base exponentiation can be computed with costs $k N_m c_m + N_s c_s$. Therefore,

$$C_{\text{sep}} = n \times (k N_m c_m + N_s c_s)$$

The dominant task of batch relation test (3) is the product of $n k_x + k_o$ exponentiations which costs

$$C_{\text{bat}} = (n k_x + k_o) N_m c_m + N_s c_s$$

In general, $V_{\text{sep}}(\mathcal{I}_n^k)$ and $V_{\text{bat}}(\mathcal{I}_n^k)$ can have different values of N_m, N_s, c_m and c_s . We ignore this effect, since N_s has only a slight effect on $V_{\text{bat}}(\mathcal{I}_n^k)$, and no significant difference is expected with respect to N_m . When G is a subgroup of \mathbb{Z}_p^* , c_m and c_s can be fixed. Even for elliptic curves, there is no significant difference between $V_{\text{sep}}(\mathcal{I}_n^k)$ and $V_{\text{bat}}(\mathcal{I}_n^k)$ [12]. Then from Definition (5), we have

$$c_{\text{gain}} = \left((k_o - 1) - \frac{k_o}{n} \right) N_m c_m + \left(1 - \frac{1}{n} \right) N_s c_s$$

Since we are only concerned with a large number of items, we assume that $k_o \ll n$ (and $1 \ll n$). Then we have

$$c_{\text{gain}} \approx (k_o - 1)N_{\text{m}}c_{\text{m}} + N_{\text{s}}c_{\text{s}} \quad (7)$$

Through the rest of this paper, we estimate the efficiency averaged among the verification items, since the above approximation is independent of n . According to Approximation (7), the criterion $c_{\text{loss}} < c_{\text{gain}}$ is written by

$$c_{\text{loss}} < (k_o - 1)N_{\text{m}}c_{\text{m}} + N_{\text{s}}c_{\text{s}} \quad (8)$$

We call this inequality ‘*Criterion*’ (8). Let q be the order of G : for almost all practical exponentiation schemes [1], we observe

$$N_{\text{m}} = \frac{1}{\omega} \lfloor \log_2 q \rfloor, \quad N_{\text{s}} = \lfloor \log_2 q \rfloor \quad (9)$$

where $2 \leq \omega$. We call factor ω the ‘*abstract window size*’, which depends on the exponentiation scheme (and q). For example, $\omega = 2$ for the binary method, $\omega = 8/3$ for the signed binary method, and will be greater for a more sophisticated method [10]. In general, $\omega \ll \log_2 q$, and if $q \sim 2^{160}$ then we observe $\omega \lesssim 5$. Suppose that $c_{\text{m}} \approx c_{\text{s}}$, then from Inequality (8) we have

$$c_{\text{loss}}/c_{\text{m}} < \left(\frac{1}{\omega} (k_o - 1) + 1 \right) \lfloor \log_2 q \rfloor \quad (10)$$

We also call this inequality Criterion (10). This criterion is the reason why lenient batch verification is more efficient than separate verification. That is if $c_{\text{loss}} = 0$, then Criterion (10) is always satisfied since $0 \leq k_o$ and $2 \leq \omega$. This means that lenient batch verification is always faster than separate verification. However, if either Approximation (7) or Observation (9) does not hold, we may have different results. We will deal with such a special case later in this paper.

Note that when $c_{\text{loss}} \approx c_{\text{gain}}$ batch verification is not recommended even if $c_{\text{loss}} < c_{\text{gain}}$, because the cost of random number generation, exponent number generation, etc. are ignored. Therefore, if $c_{\text{gain}} - c_{\text{loss}}$ is comparable to the decrease of one exponentiation base, that is

$$N_{\text{m}}c_{\text{m}} \lesssim c_{\text{gain}} - c_{\text{loss}}$$

then we ignore this matter, otherwise it is given special attention.

5 Detailed Analysis

5.1 Subgroup of \mathbb{Z}_p^* for Random Prime p

In this section, we examine the order- q subgroup of \mathbb{Z}_p^* without any special conditions of r such that $p = 2qr + 1$. We call such a prime, p , ‘*random prime*’.

¹ Instead, we assume that no fixed bases, no fixed powers, and no off-line precomputation tables are available.

We show that lenient batch verification is always faster than separate verification for this group, and investigate the conditions under which strict batch verification has an advantage over separate verification.

Let p, q be prime such that $q|(p-1)$, and let G be the order- q subgroup of \mathbb{Z}_p^* . We can use Criteria (8) and (10) in this case, which means that lenient batch verification is always faster than separate verification for this group. Therefore, we only discuss strict batch verification in this section. The ideal input of h_0 is an element of G . However, an element of G is practically represented as an element of \mathbb{Z}_p . There is an additional cost to ensure $h_0 \in G$. If p is a random prime such that $p-1$ has many small factors, there is no means except $h_0^q = 1$. Suppose that h_0^q is calculated by the window method or its derivation, then we have an approximation of c_{loss} as

$$c_{\text{loss}} \approx N_m c_m + N_s c_s$$

Then Criterion (8) is equivalent to $2 < k_o$. In this case, strict batch verification is useless unless it has three or more common exponentiation bases among all verification items. However, the gain c_{gain} is so large even if $k_o = 0$ that strict batch verification can be constructed if we find an efficient domain test for $h_0 \in G$. For example, if we use the binary method to calculate h_0^q , the loss can be estimated precisely as

$$c_{\text{loss}} = (w_H(q) - 1) c_m + \lfloor \log_2 q \rfloor c_s$$

where $w_H(q)$ is the Hamming weight of q . In this case Criterion (10) is equivalent to

$$w_H(q) - 1 < \frac{1}{\omega} (k_o - 1) \lfloor \log_2 q \rfloor$$

If we choose q as $w_H(q) \ll \lfloor \log_2 q \rfloor / \omega$, then we can relax the necessary condition for strict batch verification², that is $1 < k_o$.

5.2 Subgroup of \mathbb{Z}_p^* for Special Prime p

In this section, we examine the order- q subgroup of \mathbb{Z}_p^* on condition that all prime factors of r are greater than q . We call such a prime, p , ‘special prime’. According to the results in the previous section, lenient batch verification is always faster than separate verification in this case. Therefore, we discuss only strict batch verification. We show that strict batch verification is always faster than separate verification in this case.

Under this special condition, we can substitute $h_0^{qr} = 1$ for $h_0^q = 1$, and clearly the Legendre symbol (h_0/p) is available for this purpose, in which the computational complexity is $O((\log_2 p)^2)$. We assume that the Legendre symbol costs $\mathcal{L} \times c_m$. Thus,

$$c_{\text{loss}} = \mathcal{L} \times c_m$$

² Clearly, it is sufficient that the (quasi)optimal q -th power costs much less than c_{gain} , the necessary condition is independent of $w_H(q)$.

Then Criterion (10) is equivalent to

$$\mathcal{L} < \left(\frac{1}{\omega}(k_o - 1) + 1 \right) \lfloor \log_2 q \rfloor$$

If $p \sim 2^{1024}$, then we experimentally observe $\mathcal{L} \sim 10$ (according to GP/PARI CALCULATOR). We can satisfy this inequality for practical p, q (e.g., $p \sim 2^{1024}$, $q \sim 2^{160}$) even if $k_o = 0$ because $2 \leq \omega$. In this case, we conclude that the strict batch verification is always faster than separate verification for practical p, q .

5.3 \mathbb{F}_p -Rational Points on E/\mathbb{F}_p ($3 < \text{char } \mathbb{F}_p$)

Let \mathbb{F}_p be a finite field, we define an elliptic curve over \mathbb{F}_p as

$$E/\mathbb{F}_p : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6, \quad (a_i \in \mathbb{F}_p) \quad (11)$$

Let \mathbb{F}_{p^m} be the extension of \mathbb{F}_p of degree m . The group of \mathbb{F}_{p^m} -rational points on E/\mathbb{F}_p which we denote $E(\mathbb{F}_{p^m})$, is defined as a set of solutions $(x, y) \in \mathbb{F}_{p^m} \times \mathbb{F}_{p^m}$ on E/\mathbb{F}_p together with a point at infinity denoted by \mathcal{O} .

Groups of elliptic curves are classified by its definition field. In this section we assume $m = 1$. If p is a large prime, we call it ‘*prime field*.’ If p is a prime power such that $3 < \text{char } \mathbb{F}_p$, we call it ‘*OEK*.’ We discuss both of them here. Criterion (10) is available in this case. Therefore, we can recapitulate the results of Section 4.2 for lenient batch verification. Thus, we discuss only strict batch verification and show that it is always faster than separate verification.

Although h_0 must be an element of $E(\mathbb{F}_p)$, an element of $E(\mathbb{F}_p)$ is practically represented as $\mathbb{F}_p \times \mathbb{F}_p$. In this case, we can choose curves with a prime order, and all recommended curves over a prime field in [13] have a prime order. Here h_0 is given as $h_0 = (x, y) \in \mathbb{F}_p^2$ and clearly all that is necessary is to check if the given point satisfies Curve equation (11). It costs less than only one elliptic addition on $E(\mathbb{F}_p)$. It is sufficient for Criterion (10) if

$$1 < \left(\frac{1}{\omega}(k_o - 1) + 1 \right) \lfloor \log_2 q \rfloor$$

When $q > 4$, this will always be satisfied because we hold $0 \leq k_o$ and $2 \leq \omega$. We conclude that strict batch verification is practically always faster than separate verification in this case. Note that if this check is omitted, attackers could easily cheat the system into performing operations beyond the group, and we should acknowledge that lenient batch verification is useless for any practical cryptographic applications in this case. If we choose compressed representation (See Appendix C), there is less of an advantage for the attacker than in the case of redundant representation, but they can choose x such that we cannot solve y on \mathbb{F}_p . Some algorithms to calculate y may not detect this attack and give a false solution, in such a case we must check Curve equation (11).

5.4 Subgroup of \mathbb{F}_{2^m} -Rational Points on E/\mathbb{F}_{2^m}

On Curve equation (11), if p is a power of 2, we call the groups of elliptic curve ‘binary field’, which is the subject of this section. Criterion (10) is valid for this case, then we can apply the results of Section 4.2 for lenient batch verification. Therefore, we discuss only strict batch verification and show that it is always faster than separate verification.

In this case, it is not sufficient only to test Curve equation (11) if we want to ensure $h_0 \in G$, because all non-supersingular curves must have an even order. We can choose a curve whose order can be factorized into a 2 fold large prime and these types of curves have an effective large subgroup domain test. That is

$$T(x) \stackrel{?}{=} T(a_2) \quad (12)$$

where $T(x)$ is the *trace* of x -coordinate and $T(a_2)$ is the trace of the curve parameter a_2 . See [14] for details. All recommended random curves over \mathbb{F}_{2^m} in [13] have this type of order. Slight calculations are necessary for this (and Curve equation (11)).

The situation is the same as in Section 5.3. We conclude that strict batch verification is always faster than separate verification.

5.5 Subgroup of \mathbb{F}_{p^m} -Rational Points on E/\mathbb{F}_p ($1 < m$)

On ‘binary field’ and ‘OEF’, we can chose ‘special curves’ called ‘Koblitz curve’ [15,16] to accelerate the calculation. In this section, we discuss such special curves and we show that lenient batch verification is not always faster than separate verification in this case. Afterwards, we propose an efficient domain test and investigate the conditions under which strict batch verification becomes faster than separate verification.

Let p be prime or a prime power. Let $E(\mathbb{F}_{p^m})$ be a \mathbb{F}_{p^m} -rational point group on E/\mathbb{F}_p . Because $E(\mathbb{F}_{p^m})$ must have a small order subgroup $E(\mathbb{F}_p)$, the order of $E(\mathbb{F}_{p^m})$ is essentially a composite. We must chose the curve parameters such that the curve has no small subgroups except $E(\mathbb{F}_p)$. Therefore, m must be prime at least. All recommended curves over \mathbb{F}_2 in [13] have a composite order which can be factorized into a $\#E(\mathbb{F}_p)$ fold large prime. Let q be the large prime. Let G be an order- q subgroup of $E(\mathbb{F}_{p^m})$.

The optimal elliptic exponentiation scheme on G uses the Frobenius map ϕ which is a special constant multiplication that costs almost 0 [15,16]. And many elliptic doublings are replaced by ϕ . This does not allow us to apply Criteria (10) because Observation (9) is not true in this case. Instead of Observation (9) we should apply

$$N_m = \frac{1}{\omega} \lfloor \log_2 q \rfloor, \quad N_s = \lfloor \log_2 p \rfloor$$

in this case [15,16]. Assume that $\log_2 q \sim (m-1) \times \log_2 p$, then

$$N_s \approx \left\lfloor \frac{\log_2 q}{m-1} \right\rfloor$$

This means N_s becomes very small. If we suppose $c_m \approx c_s$ then $c_{\text{loss}} < c_{\text{gain}}$ is equivalent to

$$c_{\text{loss}}/c_m < \frac{1}{\omega}(k_o - 1) \lfloor \log_2 q \rfloor + \left\lfloor \frac{\log_2 q}{m - 1} \right\rfloor \quad (13)$$

We discuss lenient batch verification with the right side of Inequality (13) later in this section.

On the other hand we need the estimation of the loss for strict batch verification. In this case it is not sufficient to check Curve equation (11) if we want to ensure $h_0 \in G$ (however, it is still necessary). We must find an effective test to ensure that h_0 belongs to the order- q subgroup of $E(\mathbb{F}_{q^m})$. We can apply the Frobenius map ϕ for this purpose and we propose an efficient domain test³. Assume that $m > 1$ and m is relatively prime to $\#E(\mathbb{F}_p)$, then for given point P on the curve, $P \in G$ is equivalent to

$$\sum_{i=0}^{m-1} \phi^i P = \mathcal{O} \quad (14)$$

(See Appendix A). In general, the computational complexity of a Frobenius map ϕ^i is at most as much as that of a multiplication on \mathbb{F}_{p^m} , and can be much smaller if we choose a special representation for \mathbb{F}_{p^m} . Therefore, the cost of the domain test is estimated as the number of necessary elliptic additions for (14). We obtain at most

$$c_{\text{loss}}/c_m \approx \lfloor \log_2 m \rfloor + w_H(m) - 1$$

where $w_H(m)$ is the Hamming weight of m (See Appendix B). According to the overlap factor, k_o , the advantage of lenient or strict batch verification becomes apparent.

- $k_o = 0$: If $q \sim 2^{160}$ then the abstract window size, ω , is at most 5 or a similar value in this case. Therefore, if $\omega < m - 1$, then the right side of Inequality (13) will become negative. This means if $k_o = 0$, then lenient batch verification is not faster than separate verification on $E(\mathbb{F}_{2^m})$ and most of $E(\mathbb{F}_{p^m})$. Obviously strict batch verification is not faster than that.
- $k_o = 1$: If $m > \log_2 q + 1$, then the right side of Inequality (13) becomes 0, and if m is comparable to $\log_2 q$ then the gain becomes almost 0. This means lenient batch verification is slower than separate verification on $E(\mathbb{F}_{2^m})$. Furthermore, when $q \sim 2^{160}$, to satisfy Inequality (13) for strict batch verification, we need $m < 23$. The gain is so small that we do not recommend strict batch verification in this case.
- $k_o \geq 2$: The right side of Inequality (13) is always positive, and lenient and strict batch verifications are always faster than separate verification for a practical case.

³ The trace test is also available for some special case of $q = 2$, but finally the same results are obtained.

References

1. Brands, S.: Untraceable Off-line Cash in Wallet with Observers. In Stinson, D., ed.: *Advances in Cryptology — CRYPTO'93*. Volume 773 of *Lecture Notes in Computer Science.*, Berlin; Heidelberg; New York, Springer-Verlag (1993) 302–318
2. Sako, K., Kilian, J.: Receipt-Free Mix-Type Voting Scheme — A practical solution to the implementation of a voting booth —. In Guillou, L.C., Quisquater, J.J., eds.: *Advances in Cryptology — EUROCRYPT'95*. Volume 921 of *Lecture Notes in Computer Science.*, Berlin; Heidelberg; New York, Springer-Verlag (1995) 393–403
3. Abe, M.: Universally verifiable mix-net with verification work independent of the number of mix-servers. In Nyberg, K., ed.: *Advances in Cryptology — EUROCRYPT'98*. Volume 1403 of *Lecture Notes in Computer Science.*, Berlin; Heidelberg; New York, Springer-Verlag (1998) 437–447
4. Abe, M.: Mix-networks on Permutation Networks. In Lam, K., Okamoto, E., Xing, C., eds.: *Advances in Cryptology — ASIACRYPT'99*. Volume 1716 of *Lecture Notes in Computer Science.*, Berlin; Heidelberg; New York, Springer-Verlag (1999) 258–273
5. Abe, M., Hoshino, F.: Remarks on mix-network based on permutation network. In Kim, K., ed.: *Public Key Cryptography 4th International Workshop on Practice and Theory in Public Key Cryptosystems, PKC 2001*. Volume 1992 of *Lecture Notes in Computer Science.*, Berlin; Heidelberg; New York, Springer-Verlag (2001) 317–324
6. Naccache, D., M'Raihi, D., Vaudenay, S., Rphaeli, D.: Can D.S.A be Improved ? - Complexity Trade-Offs with the Digital Signature Standard -. In Santis, A.D., ed.: *Advances in Cryptology — EUROCRYPT'94*. Volume 950 of *Lecture Notes in Computer Science.*, Berlin; Heidelberg; New York, Springer-Verlag (1995) 77–85
7. Sung-Ming Yen, Chi-Sung Lai: Improved Digital Signature Suitable for Batch Verification. *IEEE Transactions on Computers* **44** (July 1995) 957–959
8. Bellare, M., Garay, J.A., Rabin, T.: Fast Batch Verification for Modular Exponentiation and Digital Signatures. In Nyberg, K., ed.: *Advances in Cryptology — EUROCRYPT'98*. Volume 1403 of *Lecture Notes in Computer Science.*, Berlin; Heidelberg; New York, Springer-Verlag (1998) 236–250
9. Boyd, C., Pavlovski, C.: Attacking and Repairing Batch Verification Schemes. In Okamoto, T., ed.: *Advances in Cryptology — ASIACRYPT2000*. Volume 1976 of *Lecture Notes in Computer Science.*, Berlin; Heidelberg; New York, Springer-Verlag (2000) 58–71
10. Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: *Handbook of applied cryptography*. CRC Press (1997)
11. Okamoto, T.: Provably Secure and Practical Identification Schemes and Corresponding Signature Schemes. In Brickell, E.F., ed.: *Advances in Cryptology — CRYPTO'92*. Volume 740 of *Lecture Notes in Computer Science.*, Berlin; Heidelberg; New York, Springer-Verlag (1993) 31–53
12. Cohen, H., Miyaji, A., Ono, T.: Efficient Elliptic Curve Exponentiation Using Mixed Coordinates. In Ohta, K., Pei, D., eds.: *Advances in Cryptology — ASIACRYPT'98*. Volume 1514 of *Lecture Notes in Computer Science.*, Berlin; Heidelberg; New York, Springer-Verlag (1998) 51–65
13. NIST: Recommended Elliptic Curves for Federal Government Use (1999) (available at <http://csrc.nist.gov/csrc/fedstandards.html/>).
14. Seroussi, C.: Compact Representation of Elliptic Curve Points over \mathbb{F}_{2^n} (April 1998) Research Manuscript, Hewlett-Packard Laboratories,.

15. Koblitz, N.: CM-Curves with Good Cryptographic Properties. In Feigenbaum, J., ed.: *Advances in Cryptology — CRYPTO'91*. Volume 576 of *Lecture Notes in Computer Science*, Berlin; Heidelberg; New York, Springer-Verlag (1992) 279–287
16. Kobayashi, T., Morita, H., Kobayashi, K., Hoshino, F.: Fast Elliptic Curve Algorithm Combining Frobenius Map and Table Reference to Adapt to Higher Characteristic. In Stern, J., ed.: *Advances in Cryptology — EUROCRYPT'99*. Volume 1592 of *Lecture Notes in Computer Science*, Berlin; Heidelberg; New York, Springer-Verlag (1999) 176–189 (A preliminary version was written in Japanese and presented at SCIS'99-W4-1.4).
17. Knuth, D.E.: *Seminumerical Algorithms*. Third edn. Volume 2 of *The Art of Computer Programming*. Addison Wesley (1997)
18. IEEE P1363/D13 (Draft Version 13): *Standard Specifications for Public Key Cryptography Annex E(Informative) Formats* (1999) (available at <http://grouper.ieee.org/groups/1363/P1363/draft.html>).

Appendix A

Why $\sum_{i=0}^{m-1} \phi^i P = \mathcal{O}$ Is Equivalent to $P \in G$?

Let ϕ be the Frobenius map on $E(\mathbb{F}_{p^m})$. $\phi - 1$ is an endomorphism of $E(\mathbb{F}_{p^m})$. According to the homomorphism theorem, we can decompose $E(\mathbb{F}_{p^m})$ into

$$\ker(\phi - 1) \oplus \text{Im}(\phi - 1)$$

Note that $\ker(\phi - 1) = E(\mathbb{F}_p)$ and we define G as $\text{Im}(\phi - 1)$, that is, for all $P \in E(\mathbb{F}_{p^m})$, there exists $P_{\ker} \in E(\mathbb{F}_p)$ and $P_{\text{Im}} \in G$ such that $P = P_{\ker} + P_{\text{Im}}$. Thus

$$\sum_{i=0}^{m-1} \phi^i P = mP_{\ker} + \sum_{i=0}^{m-1} \phi^i P_{\text{Im}} = mP_{\ker}$$

since

$$\sum_{i=0}^{m-1} \phi^i P_{\text{Im}} \in \text{Im}(\phi^m - 1) = \{\mathcal{O}\}$$

Suppose that m is relatively prime to $\#E(\mathbb{F}_p)$, then we obtain

$$P \in G \iff \sum_{i=0}^{m-1} \phi^i P = \mathcal{O}$$

Note that m must be a prime when the curve has no small subgroup except $E(\mathbb{F}_p)$. Therefore, the condition is the same as $m \nmid \#E(\mathbb{F}_p)$ for our purposes.

Appendix B

Optimal Addition Sequence of $\sum_{i=0}^{m-1} \phi^i P$

Let $a_0, \dots, a_n \in \mathbb{N}$ be an addition sequence of $m(=a_n)$ such that

$$a_0 = 1, \quad a_i = a_j + a_k, \quad (j, k < i)$$

This yields the addition sequence of $\sum_{i=0}^{m-1} \phi^i P$ given by $A_0, \dots, A_n \in E(\mathbb{F}_{p^m})$ such that

$$A_0 = P, \quad A_i = \phi^{a_k} A_j + A_k, \quad (j, k < i)$$

then $A_n = \sum_{i=0}^{m-1} \phi^i P$. This causes the computational complexity of A_n to be $n(m)$ elliptic additions, where $n(m)$ represents the length of the optimal addition chain of m . According to the binary addition chain of m , at most

$$n(m) \leq \lfloor \log_2 m \rfloor + w_H(m) - 1$$

See [17] for more a precise evaluation of $n(m)$.

Appendix C

Representation of Rational Point

In this appendix, we observe that the gain varies with the representation of a group if we use an elliptic curve.

Let h_0 be an element of $E(\mathbb{F}_p)$, which is practically represented as $\mathbb{F}_p \times \mathbb{F}_p$ or $\mathbb{F}_p \times \{0, 1\}$. We call the former ‘*redundant representation*’, and the latter ‘*compressed representation*’ [18]. If we choose redundant representation, Approximation (7) and Criterion (8) are available for cost evaluation. On the other hand, Approximation (7) is not accurate if we choose the compressed representation. We estimate the gain in the compressed representation and show that the disadvantage is very small.

Since we need y or some replacement before we perform some group operations on h_0 and given h_0 has only x (and 1 bit for y) in compressed representation, we must solve y such that it satisfies Curve equation (11) on \mathbb{F}_p . Therefore, c_{gain} becomes smaller than Approximation (7), and unfortunately Criterion (10) is not available for this case. We must find a replacement for Criterion (10).

The dominant task of solving y is a square root on \mathbb{F}_p or a similar calculation, which costs at most as much as one exponentiation on \mathbb{F}_p . In general, we may suppose that it works about 10 times faster than the elliptic exponentiation (scalar multiplication) because elliptic curve addition and doubling both require about 10 multiplications on \mathbb{F}_p . We assume that an exponentiation on \mathbb{F}_p costs about $\epsilon_m N_m c_m + \epsilon_s N_s c_s$ where $\epsilon_m, \epsilon_s \lesssim 1/10$, because $p \sim q$ in this case. In general, the smaller the characteristic of \mathbb{F}_p is, the smaller ϵ_m, ϵ_s we obtain. Thus, we should substitute

$$c_{\text{gain}} \approx (k_c - 1 - \epsilon_m) N_m c_m + (1 - \epsilon_s) N_s c_s$$

for Approximation (7). We obtain the following instead of Criterion (10)

$$c_{\text{loss}}/c_m < \left(\frac{1}{w} (k_c - 1 - \epsilon_m) + 1 - \epsilon_s \right) \lfloor \log_2 q \rfloor$$

Thus, the necessary conditions for batch verification are not so different from those of Criterion (10) except for ϵ_m and ϵ_s . This representation is more suited to communications.

Absolute Privacy in Voting

Dmitri Asonov¹, Markus Schaal², and Johann-Christoph Freytag¹

¹ Humboldt-Universität zu Berlin,
10099 Berlin, Germany

{asonov, freytag}@dbis.informatik.hu-berlin.de

² Technische Universität Berlin
D-10587 Berlin, Germany
schaal@cs.tu-berlin.de

Abstract. If nobody can prove (even in an all-against-one cooperation) that one did not vote with a particular cast, then one can claim anything about his cast even under oath, and has no fear of being caught. We consider the question of constructing a voting scheme that provides all participants with this “absolute” privacy.

We assume that half of the problem is already solved: The votes are evaluated so that only the result is revealed. Latest achievements of secure coprocessors are supposedly a justification for such a presumption.

We prove that even under the presumption that the voting reveals nothing but a result, the privacy of an individual input can withstand an “all-against-one” attack under certain conditions only.

First condition: The function that maps a set of casts to the result of voting must be non-deterministic. Second condition (paradoxically): for any set of casts any result must be possible.

1 Introduction

The problem of privacy in voting (e.g. [1]) is presumed to be a specific case of secure multiparty computations [2,3]. That is, the privacy of a voter is presumed to be preserved if the computation of votes is performed in a way that nothing but a result is revealed. This goal is trivially achievable by assuming a third trusted party and private channels between voters and the trusted party. Most of the work on improving privacy in voting is concentrated on achieving the same goal with more and more relaxed cryptographic assumptions [4,5,6,7,8,9].

We claim, that achieving the above goal is not sufficient to guarantee a voter that his vote cannot be revealed, even if underlying cryptographic assumptions hold. Namely, there is a second, independent problem: Voters, by cooperating against another voter, may reveal his vote by deducing it from their own votes and the result [1].

¹ This problem was passed over in all the previous work by assuming that the majority would never cooperate to break someone’s privacy. The problem is also present in the voting schemes with so-called unconditionally-secret ballots [7] or with information-theoretic privacy [8].

So, only if both problems are solved, a voter can be sure in his privacy *absolutely*. That is, a voter then can be sure, that even if everybody colludes against him, his vote stays private.

Clearly, if simply a sum function is used to calculate the result of voting, then the all-but-one cooperation resolves (and can prove) the vote of a victim by subtraction the sum of their votes from the result. We are interested in finding and investigating functions, that “smooth” the result in a way, that all-but-one cooperation cannot prove how the victim voted (or how the victim did not vote - more on this later). Jumping ahead, we call such functions “absolutely private voting functions” or “private voting functions” for short.

If we find such functions, we say that the voters are provided with *absolute privacy*, assuming of course, that the first problem (of calculating a result in a way, that nothing but a result is revealed) is solved too.

Any constant or, alternatively, some function unknown to the participants would be a private voting function. So we require, that any voting function we consider can not be a constant and must be officially known.

One motivation for this problem setting is to find out whether *absolute privacy* exists at all or not. Although such a privacy is recognized to be important², no work is found that deals with absolute privacy in voting.

Along with its academic interest, the “absolutely private voting” setting might be practically applicable in cases where the number of voters is small enough to consider the possibility that all might cooperate against one to break his privacy.

1.1 Preliminaries and Assumptions

Normally, the result of a voting is determined by a well-known function, that maps a set of casts to a voting result. We prove our theorems only for particular kind of functions, that we call voting functions. A voting function is defined in Sect. 1.3. A voting function can be deterministic (one set of casts refers to only one result) or probabilistic (one set of casts refers to several results with some probabilities).

Informally, we say that a voter has an absolute privacy in voting, if no cooperation can break his privacy. His privacy is assumed to be broken if some cooperation of participants may prove how the voter voted. The privacy is also assumed to be broken, if some cooperation may prove how (with what vote) the voter did not vote³. This is a privacy violation too because the voter cannot argue anymore that he voted with some arbitrary vote. We give formal definition and motivation for this kind of privacy in Sect. 1.4.

We assume that for any cooperation against any voter no information about his cast is known except the voting result. This assumption is shown to be

² It is said in [9]: “Voter privacy must be fail-safe - i.e., it must be assured even if everything fails, everyone colludes and there is a court order to reveal all election data.”

³ These two cases are the same, if two casts are possible only (like yes/no).

impossible if no special hardware is used [3]. However, this assumption may be taken seriously due to the commercially available secure coprocessors that passed FIPS 140-1 Level 4 test (the highest possible rank for a physical and logical protection) [10]. Generally speaking, such a device provides the calculation of any function in such a way that nothing but a result is revealed to several independent parties that provide inputs for this function. From the theoretical point of view, we assume a third trusted party (a secure coprocessor), that processes privately the result of a voting out of given votes.

We also assume, that the result of a voting is made public after the voting. Figure 1 demonstrates the basic architecture of the voting system that we consider.

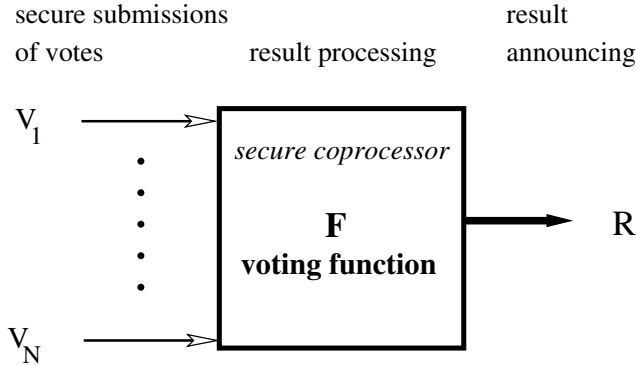


Fig. 1. Private voting using a secure coprocessor.

1.2 Our Results

We prove that:

1. Absolute privacy is not possible for deterministic voting functions (Sect. 2, Theorem 2). That is, if one wants to have only one result possible for the given casts, one never gains absolute privacy in voting.
2. Absolute privacy is possible for probabilistic voting functions (Sect. 3, Theorem 3). We give an example. Simply speaking, we show how to conduct voting such that every voter has absolute privacy.
3. All (probabilistic) voting functions that preserve absolute privacy have a well-recognized drawback (Sect. 3, Theorem 4). Namely, all results must be assigned to every set of casts with non-zero probabilities, i.e., for any set of casts any result is possible.

1.3 Voting Function

The voting function is formally defined as a function, that satisfies the properties listed below. These three properties (we call them influence, commutativity and openness) are used later to prove our results.

By N we denote the number of voters. R denotes the voting result. The (values of) casts of participants are denoted by v_1, \dots, v_N . The arguments of the function represent the casts of voters and are the values from the set V , the number of elements in the set is $|V|$. The number of possible results is denoted by $|R|$. For short, below we write $\forall v_x$ instead of $\forall v_x \in V$.

Property 1: Influence. Voters have an *influence* on the result of voting, i.e.

$$\forall v_1, \dots, v_N \quad \exists j \leq N, v'_1, \dots, v'_j \quad : \quad F(v_1, \dots, v_N) \neq F(v'_1, \dots, v'_j, v_{j+1}, \dots, v_N) \quad (1)$$

This property basically means, that a constant is not a good function for voting. On the other hand, it does not state that one vote changes result. Instead, it only states that some group of votes can change the result.

Property 2: Commutativity. By this property we require F to be commutative, i.e.

$$\forall v_1, \dots, v_N, 1 \leq i < j \leq N \quad : \quad F(v_1, \dots, v_i, \dots, v_j, \dots, v_N) = F(v_1, \dots, v_j, \dots, v_i, \dots, v_N) \quad (2)$$

One might think that this does not reflect voting models, where some voters have a privilege of a stronger impact on the result. The commutative property does not reject such models. It only states that the result does not differ if inputs are processed in different order.

Property 3: Function Openness. By this property we state, that a function definition is known⁴. This property allows us to presume in the proofs, that an all-but-one cooperation (that, of course, may include organizers) knows how to calculate the result of the function for any given input.

$$\text{One knows } F(v_1, \dots, v_N) \text{ for any } v_1, \dots, v_N \quad . \quad (3)$$

Evidently, if nobody knows how the result of a voting is processed, the voting result does not carry any information. And so this is not a voting at all.

Definition 1 (Voting Function). A function is a voting function, iff it satisfies the properties [1](#), [2](#), [3](#).

⁴ This is a voting scheme property which corresponds to a function used for the result processing. Therefore this property cannot be expressed as a formal mathematical property of a function. This property might be defined as a voting scheme property, but this would change neither the results nor the proofs. So we call it “a voting function property” for the sake of a convenient presentation.

1.4 Private Voting Function

In our definition of absolute privacy, any cooperation against one voter cannot prove that the voter did not vote with a particular value.

Imagine you have participated in a voting where you are interested in keeping your cast *absolutely* private. And one can prove that you did not vote C given that only three casts were allowed (A, B, C) . It would be natural for you to consider this as your privacy violation. There is a more specific example of why our definition of absolute privacy is appropriate. It allows a voter to claim that he voted with any arbitrary cast (even under oath), while having no fear of being caught.

Herein “absolute privacy” and “absolutely private” are often reduced to the words “privacy” and “private”. In this paper, we do not consider any privacy, except absolute one.

Definition 2 (Private Voting Function). *A voting function F is private, iff for any inputs v_1, \dots, v_N , given the first $N - 1$ inputs and the result $R = F(v_1, \dots, v_N)$, for any $A \in V$ it is impossible to prove that $v_N \neq A$.*

Example 1. This example shows what a private voting function might look like. Imagine 100 voters, each votes “1” or “0”. A function F summarizes the votes and maps the sum to a number from 0 to 9: If the sum is less or equal 10, then the result of voting is 0; if the sum is more than 10 but less or equal 20, then the result of voting is 1 and so on. The function F satisfies all properties of a deterministic voting function.

Suppose, only 15 of the 100 voters vote with “1”. Therefore the sum of the votes is 15, and the function F , by definition, produces result $R = 1$. Can any 99 voters in cooperation say something about the cast of the 100-th voter? The answer is no. Because, no matter, what was the vote of the 100-th voter (“0” or “1”), the result would not change. And, by our presumption, nothing but a result is revealed.

Still, the voting function F is not private, because it does not preserve privacy for arbitrary input. Let the number of those who voted with “1” be 10. Then the result of voting is 0. Now, consider any voter who voted with “0”. The cooperation of the rest 99 voters can tell something about his vote. More precise, in this example, they can prove that he did not vote with “1”, because if his cast were “1”, then the result would change to 1. \square

2 Deterministic Voting Functions

First, we prove a theorem about a property any private deterministic voting function must have. Second, we prove that there are no deterministic voting functions that have that property.

But first, let us formally mention, that we consider deterministic voting functions:

$$\nexists \quad v_1, \dots, v_N, \quad : \quad F(v_1, \dots, v_N) \neq F(v_1, \dots, v_N) \quad (4)$$

Theorem 1 (The Necessary Condition for a Private Deterministic Voting Function). *A voting function F is private only if*

$$\nexists \quad v_1, \dots, v_N, v'_N \quad : \quad F(v_1, \dots, v_N) \neq F(v_1, \dots, v'_N) \quad (5)$$

This is equivalent to

$$\forall \quad v_1, \dots, v_N, v'_N \quad : \quad F(v_1, \dots, v_N) = F(v_1, \dots, v'_N)$$

Proof. Suppose, from contradiction, that some voting function F is private, and (5) does not hold:

$$\exists \quad v_1, \dots, v_N, v'_N \quad : \quad F(v_1, \dots, v_N) \neq F(v_1, \dots, v'_N)$$

Let the casts be exactly v_1, \dots, v_N . So, the last equation is true for this situation:

$$\exists \quad v'_N \quad : \quad F(v_1, \dots, v_N) \neq F(v_1, \dots, v'_N)$$

This means that $N - 1$ cooperating voters, who know their own votes v_1, \dots, v_{N-1} and the result of the voting R , can (due to (3)) compute $F(v_1, \dots, v_{N-1}, x_N)$ for all the range of the values of a vote x_N . This way they find such an $x_N = v'_N$, that

$$F(v_1, \dots, v'_N) \neq R$$

From the property (4) of a deterministic function they conclude, that $v'_N \neq v_N$. That is, they can prove that the N -th voter did not vote with v'_N value. (If there are only two possible values for votes, like “yes” or “no”, they can even tell exactly what his vote was.)

Formally, the coalition of $N - 1$ voters has the right to conclude, that

$$v_N \neq v'_N$$

Then, by definition (2) F is not a private function. It is a contradiction. \square

Theorem 2 (Nonexistence of Private Deterministic Voting Function). *A deterministic voting function cannot be private.*

Proof. Assume that a private deterministic voting function F exists. Then the properties of a deterministic voting function (statements (1), (2), (3), (4)) are true for F , and F satisfies the necessary condition of a private deterministic voting function (Theorem (1)). Starting with this, we lead to a contradiction.

Let the casts of the voters be v_1, \dots, v_N and let the result be R :

$$F(v_1, \dots, v_N) = R$$

STEP1. We can find the minimal number of voters j , that would change result by changing their casts. Due to the voting function property (1), this number exists and it is less then or equal to N . Due to Theorem (1) this number must be more than 1:

$$\exists \quad 1 < j \leq N \quad : \quad \begin{array}{l} \text{Theorem } \textcolor{red}{1} \\ \text{Property } \textcolor{red}{1} \end{array}$$

$$\forall \quad v_1^*, \dots, v_{j-1}^* \quad : \quad F(v_1^*, \dots, v_{j-1}^*, v_j, v_{j+1}, \dots, v_N) = R \quad (6)$$

$$\text{And } \exists \quad v'_1, \dots, v'_{j-1}, v'_j \quad : \quad F(v'_1, \dots, v'_{j-1}, v'_j, v_{j+1}, \dots, v_N) \neq R \quad (7)$$

STEP2. In (6), let us take $v_1^* = v'_1, \dots, v_{j-1}^* = v'_{j-1}$:

$$F(v'_1, \dots, v'_{j-1}, v_j, v_{j+1}, \dots, v_N) = R \quad (8)$$

By combining (7) and (8), we get:

$$F(v'_1, \dots, v'_{j-1}, v_j, v_{j+1}, \dots, v_N) \neq F(v'_1, \dots, v'_{j-1}, v'_j, v_{j+1}, \dots, v_N)$$

In the last expression, if we take into account property 2 of a voting function, then we get a contradiction to Theorem 1. \square

3 Probabilistic Voting Functions

We considered deterministic functions. Probabilistic functions are more general functions in the sense, that for unique input they make possible different results with different probability.

In our notation, a probabilistic function is defined by assigning a finite set of result-probability pairs for each possible set of casts (see Fig. 2). Considering the example where the result is calculated inside a secure coprocessor, a secure coprocessor simply outputs the result in accordance to given probability distributions.

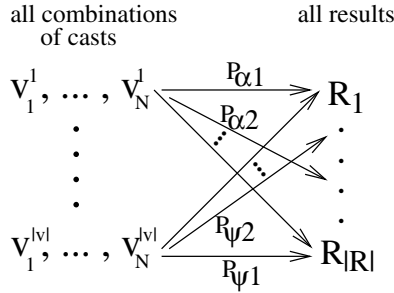


Fig. 2. Visual representation of a probabilistic function.

In this section we prove, although private voting is possible with probabilistic functions, any result is possible too. This might be viewed as a significant drawback for those who “make some use” of a result of an absolutely private voting.

From now on, whenever we refer to a voting function, or a private voting function, we undermine a probabilistic voting function or a private probabilistic voting function, if not specified exactly.

3.1 Voting Function

First, we rewrite shortly the definitions of voting function properties because the notations are slightly changed to carry the notion of probability.

Influence. Voters have an *influence* on the result of voting, i.e.

$$\forall v_1, \dots, v_N, R \quad \exists j \leq N, v'_1, \dots, v'_j \quad : \quad P(R|v_1, \dots, v_N) \neq P(R|v'_1, \dots, v'_j, v_{j+1}, \dots, v_N) \quad (9)$$

This property means, that if the distribution is the same for any casts, this probabilistic function is not good for voting.

Commutativity. Voters have an *equal* influence on the result of voting, i.e.

$$\forall v_1, \dots, v_N, R, 1 \leq i < j \leq N \quad : \quad P(R|v_1, \dots, v_i, \dots, v_j, \dots, v_N) = P(R|v_1, \dots, v_j, \dots, v_i, \dots, v_N) \quad (10)$$

Function Openness. Voters (or at least organizers) know how the votes are counted, i.e.

$$\text{One knows } P(R|v_1, \dots, v_N) \text{ for any } R, v_1, \dots, v_N \quad . \quad (11)$$

3.2 Private Voting Function

We continue using the same definition of a private voting function as in the deterministic case (Definition 2).

Example 2. Let us start with the deterministic example given in Sect. 1. We might try to fix the problem in that example in the following way:

1. If the sum is not critical for privacy (like 1,2,3,4,5,6,7,8,9,12,13,14,15,...), we output the result as before: No privacy is revealed as we discussed. (Because, for these sums, changing one vote does not change the result.)
2. If the sum is critical (like 10 or 11), we run some probabilistic function, that outputs results (like 0 or 1) with equal probability.

Suppose 11 voters voted with “1” and the result announced is 1, and 99 voters cooperate against the one (who voted with “1”). They cannot argue, that his vote was “1” (as they do in the deterministic example) because he could voted with “0” (with the same probability), then the sum would be 10, but then it would be flipped to the result 1 due to our new voting function for critical sums.

Still, the described probabilistic voting function does not preserve privacy. Counterexample: Again, consider 11 voters who voted with “1” and the result announced is 0, and 99 voters cooperate against the one (who voted with “0”). They can argue, that his vote was not “1”, because if it were “1”, then the sum would be 12 and the result 0 would be impossible.

Below we prove formally, that the private voting is still possible, but (considering this example) only if sometimes (very rarely) for the all 100 votes “1” the result 0 appears, and (also very rarely) for the all 100 votes “0” the result 9 appears. \square

Theorem 3 (Existence of a Private Probabilistic Voting Function).
There exists a private probabilistic voting function.

Proof. To prove the theorem, we give an example of a private probabilistic voting function. In other words, we give an example of a function, that satisfies all three voting function properties listed before (statements [9](#), [10](#), [11](#)) and that preserves absolute privacy of any voter (Definition [2](#)).

There are 3 voters, two possible casts (0/1 or yes/no) and 4 different results (Fig. [3](#)).

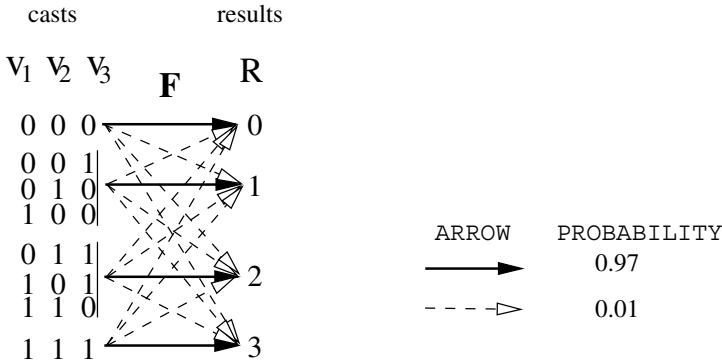


Fig. 3. An example of a private probabilistic voting function.

Clearly, the scheme might be generalized for any N , $|V|$ and $|R|$.

It also may be slightly adjusted so, that the less the result corresponds to the set of casts, the smaller probability it has (In our case, for simplicity, all such probabilities are 0.01). \square

It is remarkable, that removing any single arrow makes the function not private. This is the subject of the next theorem.

The next theorem might be seen as a necessary condition for a function to be a private probabilistic voting function. It also shows how “poor” the voting functions must represent the result of a voting in order to provide absolute privacy. Roughly speaking, it proves, that nobody can give an example of an absolutely private voting function, where for all 100 “yes” votes the result “no” is impossible.

Theorem 4 (The Flaw of a Private Probabilistic Voting Function). *Any private probabilistic voting function F has the following property:*

$$\forall R, v_1, \dots, v_N \quad : \quad P_{F(v_1, \dots, v_N)=R} \neq 0 \quad (12)$$

Proof. We give a very short proof, although it might be extended for better understanding.

Let us consider possible results (the probabilities are not zero) for the set of votes v_1, \dots, v_N . We take one of this (possible) results - R . Starting with this, we prove, that R has a non-zero probability for any other set of votes. This proves the theorem.

So, we have

$$P(R|v_1, \dots, v_N) \neq 0$$

Then let us change one of the votes in the set to the arbitrary value. Let it be $v_N \rightarrow v'_N$. Then we can write

$$P(R|v_1, \dots, v'_N) \neq 0$$

If we cannot write the last equation, then, considering the equation before the last, we have absolute privacy definition violation, if the first $N - 1$ voters cooperate against the N -th.

Using the same technique, by changing votes one by one, we achieve

$$P(R|v'_1, v'_2, \dots, v'_N) \neq 0$$

□

4 Related Work

Nothing in the related work tackles the problem setting we consider.

4.1 Secure Multi-party Computation

Secure multi-party computation (SMPC) deals with computing *securely* any probabilistic function in a distributed network where each participant holds one of the inputs. “Securely” means that no more information is revealed to a participant than can be computed from that participant’s input and a result [3, 2]. Thus SMPC does not consider the question of how much information about a single input can be revealed in the result given the other inputs. Our work might be seen as a partial answer to this question.

4.2 Electronic Elections

In the research work on security and privacy in electronic elections (see [6, 7, 11] for example), an enormous number of voting schemes have been proposed. The similarity between all those schemes is that in none of them the voter’s privacy

withstands an all-against-one attack. There is only one exception: Stochastic anonymous voting is discussed later in this section.

It seems that the hypothetical question of what it costs to protect the privacy of a single voter (or can it be protected at all) if all-but-one voters cooperate is not pondered even once.

Receipt-Free Voting. To stop vote buying, receipt free voting [1112] schemes are proposed that prevent (under such assumptions as “private booth” and “private channel” only) a voter from proving his cast to somebody else. So the all-against-one conspiracy of voters still may *know* the victim’s vote, but they cannot *prove* it to somebody else. Because in order to prove it they should prove their own votes, that is made impossible by the receipt-freeness. Again, the question is not considered how a single vote might be protected if all other votes are somehow made known.

Stochastic Anonymous Voting. In [13] a stochastic voting protocol is proposed with the main idea that the voters have to randomize their votes before submitting them. A technique is also proposed of how do organizers force users to randomize their votes. Then, even if all votes are made known, the voters’ privacy is preserved.

In that protocol, for any set of votes, any result is possible. However the question is not considered whether there exists such an absolutely private voting scheme, that not all results are possible for any given set of votes.⁵ Instead, some statistical properties of the protocol are investigated. And the primary result of that work is that the accuracy of the voting result improves as the number of voters increases.

Real Systems. There are a lot of voting systems implemented and offered by commercial companies. Some of them build their advertising campaigns on terms like “absolute privacy”⁶. What they probably mean is privacy under the assumption that all-but-one conspiracy of voters is impossible.

4.3 Statistical Disclosure Control

The problem for a statistical database is how to preserve the privacy of individual data from the anyone, who wishes to get some statistics calculated on the set of several individual data [14]. The absolutely private voting problem is different, since not only the result of the processing is known, but also all except one individual data are known too.

⁵ And the main achievement of our work is that we give and prove the *negative* answer to this question.

⁶ “Our technology provides for absolute privacy of an individual’s ballot...” is officially claimed by one electronic voting company. Yet another company claims to provide “fail-safe” voter privacy, where fail-safe means that one cannot link a voter to a vote even if everyone colludes etc.

4.4 One-Way Hash Functions

One-way hash functions produce the result in a way that, given the result and not given some (even relatively small) part x_i of the input X , it is computationally difficult to say what is this hidden input x_i equal to [15]. From the first point of view, such functions might be considered as private voting functions. They are not, because for one-way functions, although it is difficult to find $A : x_i = A$, it costs nothing to find $A : x_i \neq A$. And the latter is a privacy violation in case of voting.

4.5 Theories of Voting

The mathematics of voting (or, the theory of voting), in spite of being rather developed [16], has nothing relevant to the problem we consider. One of the corner goals of the theory is to calculate votes so, that the winner (one of more than two candidates) is “whom the voters really want” [17]. In this context, an impossibility for some particular electoral systems [18] is just one of the paradoxes known to the voting theory [19,20].

A probabilistic voting theory [21] has a goal different from the voting theory. Probabilistic voting theory is the mathematical prediction of candidate behaviour in, or in anticipation of, elections in which candidates are unsure of voters’ preferences. This theory, as well as similar ones – spatial theory of voting [22,23] and a unified theory of voting [24], does not consider privacy questions.

4.6 No Voting – No Problem

Some work was dedicated to show that an election, as a main instrument of democracy, has many disadvantages. The basic one is that after the election occurs, a winner holds the promises no more. So, the election is just an illusion.

These disadvantages of an election might be removed just by avoiding elections. Interestingly, one insists that democracy is still well possible without elections [25]. Instead of elections, other techniques are proposed to make decisions in the society.

Some of those techniques, like referendums⁷, have the same mathematical model as elections, so the privacy problem remains. Other alternatives are completely different from normal elections, so that the notion of privacy is meaningless for them. One of the most radical alternatives (the sortition) is to take randomly one of all allowed outcomes as a result [26].

5 Discussion

In this section we clarify some questionable points of the paper.

⁷ Instead of electing politicians who then make policy decisions, these decisions are made directly by the public voting [25].

5.1 The Absolutely Private Voting Scheme Proposed

From some prospective, using the absolutely private voting function we proposed in Theorem 3 is not very different from picking candidates at random. And so, one might conclude the proposed voting scheme is useless.

The proposed voting scheme is a side-effect of our work. So we do not discuss much about whether this system is good for voting or not - this is done very well by Kikuchi et al. [13] using the probability theory. The main point of our paper is a proof of the paradoxical statement: If one wants to build an absolutely private voting system, he can do nothing better than (accidentally very similar) our or Kikuchi et al. [13] scheme.

5.2 The Definition of Voting Function

Our definition of a voting function is very general. Basically, it is any non-constant function. The question is whether we should define it more strictly. We think that the more general the functions we consider to prove the theorems, the more universal these theorems are.

For example, we could improve the definition of a voting function by saying that the voting function should indicate undoubtedly if the majority (or generally, a given amount) of voters voted for a proposition. But it follows directly from Theorem 4, that such improved voting functions can not be private. It is also evident that if we substitute “undoubtedly” for “with high probability” then such private functions exist.

6 Future Work

An open issue in this work is what one can say about the privacy of a single vote, if at most $N-2$ (or $N-k$) participants are allowed to cooperate.

The work describing applications of our results for some practical Private Information Retrieval schemes (like one described in [27]) is in progress.

7 Conclusion

Voting is absolutely private if a voter can insist that he voted with an arbitrary cast (to preserve his privacy) and nobody (in any conspiracy) can prove that he lies.

Our goal was to investigate whether or not absolute privacy is possible. And if it is possible, then under which conditions.

We considered special types of functions, that are likely to reflect the most important properties of any voting.

For these functions, we proved that absolute privacy holds only if the voting function is probabilistic and for any set of casts any result is possible.

In summary, absolute privacy has a price. It is up to the participants to decide whether they want the “real” result or “real” privacy. It is impossible to

have both simultaneously⁸. We have shown that there is a tradeoff between the absolute privacy of a vote and the precision of a result of the voting.

Acknowledgements. Many thanks go to Myra Spiliopoulou for the valuable suggestions on the terminology and the context, and to Bernhard Thalheim for an interesting discussion about related work, and to anonymous referees for valuable notes and recommendations. Additional comments were kindly given by Hans-Joachim Lenz, Oliver Günther, Johannes Köbler, and Rainer Conrad.

This research was supported by the German Research Society, Berlin-Brandenburg Graduate School in Distributed Information Systems (DFG grant no. GRK 316).

References

1. Cramer, R., Gennaro, R., Schoenmakers, B.: A secure and optimally efficient multi-authority election scheme. In: *Theory and Application of Cryptographic Techniques*. (1997) 103–118
2. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or a completeness theorem for protocols with honest majority. In: *Proceedings of STOC'87*. (1987)
3. Goldreich, O.: Preface to special issue on general secure multi-party computation. <http://www.wisdom.weizmann.ac.il/~oded/PS/preSI.ps> (1999)
4. Chaum, D.: Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM* **24** (1981) 84–88
5. Cohen, J., Fischer, M.: A robust and verifiable cryptographically secure election scheme. In: *Proceedings of 26th FOCS*. (1985)
6. Cohen, J.: Improving privacy in cryptographic elections. Technical Report 454, Yale University, Department of Computer Science (1986)
7. Chaum, D.: Elections with unconditionally-secret ballots and disruption equivalent to breaking RSA. In: *Advances in Cryptology: Proc. of EuroCrypt'88*, LNCS 330, SpringerVerlag. (1988) 177–182
8. Cramer, R., Franklin, M., Schoenmakers, B., Yung, M.: Multi-authority secret-ballot elections with linear work. In: *Proceedings of EUROCRYPT'96*, LNCS 1070. (1996)
9. Gerck, E.: Internet voting requirements. *The Bell* **1** (2000) 3–5,11–13
10. Smith, S.W., Palmer, E.R., Weingart, S.H.: Using a high-performance, programmable secure coprocessor. In: *Proceedings of the 2nd International Conference on Financial Cryptography*, Springer-Verlag LNCS. (1998)
11. Benaloh, J., Tuinstra, D.: Receipt-free secret-ballot elections. In: *Proceedings of the 26th ACM Symposium on Theory of Computing*. (1994) 544–553
12. Hirt, M., Sako, K.: Efficient receipt-free voting based on homomorphic encryption. In Preneel, B., ed.: *Advances in Cryptology - EUROCRYPT'00*. Volume 1807 of *Lecture Notes in Computer Science*, Springer-Verlag (2000) 539–556

⁸ This might be seen as a self-contradiction of democracy: Privacy and voting are two important aspects of democracy. But we have shown that they cannot perfectly coexist, i.e., one or the other must inherently be flawed.

13. Kikuchi, H., Akiyama, J., Gbioff, H., Nakamura, G.: Stochastic anonymous voting. Technical Report CMU-CS-98-112, Carnegie Mellon University (1998)
14. Willenborg, L., de Waal, T.: Statistical Disclosure Control in Practice. Volume 111 of Lecture Notes in Statistics. Springer-Verlag (1996)
15. Schneier, B.: Applied Cryptography. 2nd edn. Wiley, New York (1996)
16. Saari, D.G.: Basic Geometry of Voting. Springer-Verlag (1995)
17. Saari, D.G.: Geometry, voting, and paradoxes. *Mathematics Magazine* (1998) 243–259
18. Woodall, D.R.: An impossibility theorem for electoral systems. *Discrete Mathematics* (1987) 209–211
19. Saari, D.G.: A dictionary for voting paradoxes. *Journal of Economic Theory* (1989) 443–475
20. Nurmi, H.: Voting Paradoxes and How to Deal with Them. Springer-Verlag (1999)
21. Coughlin, P.J.: Probabilistic Voting Theory. Cambridge University Press (1993)
22. Enelow, J.M., ed.: Spatial Theory of Voting. Cambridge University Press (1984)
23. Enelow, J.M., Hinich, M.J., eds.: Advances in the Spatial Theory of Voting. Cambridge University Press (1990)
24. Samuel Merrill, I., Grofman, B.: A Unified Theory of Voting. Cambridge University Press (1999)
25. Martin, B.: Democracy without elections. *Social Anarchism* (1995-96) 18–51
26. Carson, L., Martin, B.: Random selection in politics. Praeger (1999)
27. Smith, S.W., Safford, D.: Practical private information retrieval with secure coprocessors. Technical report, IBM Research Division, T.J. Watson Research Center (2000)

A Logical Model for Privacy Protection

Tsan-sheng Hsu, Churn-Jung Liao, and Da-Wei Wang

Institute of Information Science
Academia Sinica, Taipei, Taiwan
{tshsu,liaucj,wdw}@iis.sinica.edu.tw

Abstract. In this paper, we present a logical model for privacy protection problem in the database linking context. Assume in the data center, there are a large amount of data records. Each record has some public attributes the values of which are known to the public and some confidential attributes the values of which are to be protected. When a data table is released, the data manager must assure that the receiver would not know the confidential data of any particular individuals by linking the releasing data and the prior information he had before receiving the data.

To solve the problem, we propose a simple epistemic logic to model the user's knowledge. In the model, the concept of safety is rigorously defined and an effective approach is given to test the safety of the released data. It is shown that some generalization operations can be applied to the original data to make them less precise and the release of the generalized data may prevent the violation of privacy. Two kinds of generalization operations are considered. The level-based one is more restrictive, however, a bottom-up search method can be used to find the most informative data satisfying the safety requirement. On the other hand, the set-based one is more flexible, however, the computational complexity of searching through the whole spaces of this kinds of operations is much higher than the previous one though graph theory is used to simplify the discussion. As a result, heuristic methods may be needed to improve the efficiency.

Keywords: Privacy, Data table, Epistemic logic.

1 Introduction

With the rapid development of computer and communication technology, it has become much easier to store massive amount of data in a central location and spread them to the end users via Internet. In appropriate use, these data may be valuable information sources for scientists, analysts, or policy and decision makers. However, there may be a great danger of privacy invasion if they are accessed without any restriction. As in [Cam00], "in the past, most individuals lacked the time and resources to conduct surveillance required to invade an individual's privacy, as well as the means to disseminate the information uncovered, so privacy violations were restricted to those who did, mainly the government

and the press". However, the development of Internet technology has changed the situation radically. Nowadays, any individual Internet users may easily spread a piece of information to the worldwide audience within seconds. Under such situation, the revelation of private information to the unauthorized users, even not intentionally, may cause serious invasion of human rights.

There are many technical problems to be addressed for privacy protection. Though the authorization and authentication mechanisms can prevent illegal access of the databases to a large extent, it is also the responsibility of the data center to ensure that the users can not infer privacy information from the legally received data. This is traditionally called inference control problem in database security [CHKW00, Den82, Mor88]. In particular, it should be avoided that the users can know the private information of an individual by linking some public or easy-to-know database with the data they receive legally from the data center.

On the other hand, the over-restriction of access to the database may render the data useless. Thus the main challenge is to achieve the balance between the privacy and data availability. To achieve the purpose, the database manager must check what kind of knowledge can be derived from the to-be-disclosed data before they respond to the users' queries. Under the restriction of no privacy violation, the more knowledge the data can derive, the more useful it is for the users. According to the checking results, the database managers may decide to refuse the user's query or respond to it with some modification of the data. Therefore, the modeling of the user's knowledge becomes of primary importance for the privacy protection work.

In artificial intelligence and distributed computing, epistemic logic has played the main role in the modeling of knowledge [EHMV96], so it is natural to adopt the framework for the current purpose. However, for some technical reasons to be made explicit later, the two-valued epistemic logic cannot be applied directly. Instead, we will need a partiality of truth values, so in this paper, the privacy protection problem is modeled in a partial epistemic logic framework. The semantics of the logic is analogous to that for partial modal logic [JT96]. However, since the knowledge about knowledge (i.e., the so-called introspective knowledge) is not our concern, nested modalities are not needed in the current framework, so the syntax and semantics can be slightly simplified.

The partial epistemic logic model provides us a rigorous definition of privacy violation. When it is found that a direct release of a data table may cause privacy invasion, what should the database manager do? One approach is just to refuse the release of the data, which is definitely too conservative to be useful. Thus an alternative approach is to modify the data before they are released. A common modification method is to generalize the values of some data cells to a coarser level of precision. This approach, when used appropriately, will make it impossible to identify the private information of any individuals but the user can still induce useful general knowledge from the data [Swe97]. For the generalization of the data values, we can partition the domain of values according to some levels of precision and try to generalize the data from the finest to the coarsest level until the privacy condition is met. This kind of operation is

called level-based generalization. On the other hand, we can try to merge some values in the data and replace the single value by the merged result. Since the generalized values may be any nonempty subset of the domains, this process is called set-based generalization. Obviously, set-based generalization provides more flexibility in avoiding privacy violation while keeping valuable information.

In the rest of the paper, we will first formally state the privacy protection problem in the database linking context. In Section 2 a simple partial epistemic logic is used for modeling the situation when only level-based generalization is allowed in the modification of the data. The simple framework is then further generalized in Section 3 to handle the case where set-based generalization operations are also allowed. Finally, some further research directions will be discussed in Section 4.

1.1 The Privacy Protection Problem

To state the privacy protection problem in a database linking context, we must first fix the data representation. The most popular data representation is by data table [Paw91]. The data in many application domains, for example, medical records, financial transaction records, employee data, and so on, can all be represented as data tables. A formal definition of data table is given in [Paw91].

Definition 1 A data table¹ is a pair $T = (U, A)$ such that

- U is a nonempty finite set, called the universe,
- A is a nonempty finite set of primitive attributes, and
- every primitive attribute $a \in A$ is a total function $a : U \rightarrow V_a$, where V_a is the set of values of a , called the domain of a .

A relational database in general consists of a number of data tables and a data table may contain more records than those for U . However, we will only concern the privacy protection problem in one interactive session between the user and the data center. In such interactive session, the user asks the data center to release the data about a given set of individuals U , so without loss of generality, we can consider the sub-table consisting of only records of those individuals.

The attributes of a data table can be divided into three sets. The first one contains the *key attributes*, which can be used to identify to whom a data record belongs, so they are always masked off in response to a query. Since the key attributes uniquely determine the individuals, we can assume that they are associated with elements in the universe U and omit them from now on. Second, we have a set of *public attributes*, the values of which are known to the public. For example, in [Swe97], it is pointed that some attributes like birth-date, gender, ethnicity, etc., are included in some public databases such as census data or

¹ Also called knowledge representation system, information system, or attribute-value system

voter registration lists. These attributes, if not appropriately generalized, may be used to re-identify an individual's record in a medical data table, and this will cause privacy violation. The last kind of attributes is the *confidential ones*, the values of which are mainly the goals we have to protect. Sometimes, there is an asymmetry between the values of a confidential attribute. For example, if the attribute is the HIV test result, then the revelation of a '+' value may cause serious privacy invasion, whereas it does not matter to know an individual have a '-' value.

To separately handle the three kinds of attributes, a data table can be re-organized as a data matrix and a mapping from the universe to the rows of the matrix. A data matrix \mathbf{T} is a $n \times m$ matrix $[t_{ij}]_{n \times m}$ such that for each $1 \leq j \leq m$, $t_{ij} \in V_j$, where V_j is the domain of attribute j . The data matrix has n records and each record has m attributes. We denote the row vectors of \mathbf{T} by $\mathbf{t}_1, \dots, \mathbf{t}_n$. Let $m = m_1 + m_2$, where attributes $1, \dots, m_1$ are the public ones and $m_1 + 1, \dots, m$ the confidential ones, then $p(\mathbf{T})$ and $c(\mathbf{T})$ denotes respectively the sub-matrix consisting of the first m_1 columns and the last m_2 columns of \mathbf{T} . Analogously, each row vector \mathbf{t} can also be cut into two parts, $p(\mathbf{t})$ and $c(\mathbf{t})$.


The information that a data center has is a triple (U, \mathbf{T}, ι) , where U is a set of individuals (the universe), \mathbf{T} is a data matrix as defined above, and $\iota : U \rightarrow \{\mathbf{t}_1, \dots, \mathbf{t}_n\}$ is an identification mapping which assigns to each individual a data record (i.e., a row in the data matrix). The information the user has is another triple $(U, p(\mathbf{T}), \iota_p)$, where U and $p(\mathbf{T})$ are defined as above and $\iota_p : U \rightarrow \{p(\mathbf{t}_1), \dots, p(\mathbf{t}_n)\}$ is the identification mapping known by the user. We assume the user's information about the public attributes is correct, so $\iota_p(u) = p(\iota(u))$ for any $u \in U$. From now on, we will fix the context with (U, \mathbf{T}, ι) and $(U, p(\mathbf{T}), \iota_p)$. Thus the privacy protection problem is

How can \mathbf{T} be modified and then sent to the user so that the user would not know any individual's confidential attribute values and the modified matrix is kept as informative as possible?

To solve the problem, we have to answer precisely the following three questions.

1. What kind of operations can be applied to modify the matrix?
2. What does it mean that the user knows an individual's confidential information?
3. What is the meaning of a matrix being more informative than the others?

In the following two sections, we will try to answer the questions in an epistemic logic framework. However, before proceeding to the formal definitions, let us explain the basic ideas by an example.

Example 1  Assume the data center has the following 8×2 data matrix for individuals u_1, \dots, u_8 , where the first attribute denotes the height of patients and the second one is the test result of some disease.

² The example is simplified from one in [\[Chi00\]](#).

	height	disease
u_1	160	+
u_2	160	+
u_3	160	−
u_4	165	+
u_5	170	−
u_6	170	−
u_7	175	+
u_8	175	+

Suppose only the first attribute is the public one, so the user has an identification mapping ι such that $\iota(u_1) = \iota(u_2) = \iota(u_3) = 160$, $\iota(u_4) = 165$, $\iota(u_5) = \iota(u_6) = 170$, and $\iota(u_7) = \iota(u_8) = 175$. Obviously, if the matrix is sent to the user without any modification, then the privacy of u_4 is invaded. This is due to the uniqueness of the height of u_4 in the public part of the data table.

In [Swe97], the notion of *bin size* is proposed to resolve the problem. A *bin* is defined as an equivalence class according to the public attributes and bin size is its cardinality. Since the bin size for height 165 is equal to 1, so it is not safe for privacy protection. Thus it is required that a safe table must satisfy the condition that the size of any bin is greater than 1. However, even the size for the bin with height 175 is 2, the user can still infer that both u_7 and u_8 have the disease since they have the same confidential attribute values. Hence from the epistemic perspective, the protection by bin size is not adequate. Though in general, the chance for the user to know the confidential information is smaller if the bin size is larger, it has been well-known that controlling bin size alone is not sufficient to stop inference attacks [Den82]. To properly protect the privacy, the data center may replace the heights 160 and 165 in the matrix by an interval value $[160,165]$ and the heights 170 and 175 by $[170,175]$ and then send the modified matrix to the user. \square

2 Logical Model for Level-Based Generalization

2.1 The Generalization Operations

From Example 2, it can easily be seen that the main operations for modifying the matrix is to replace the public attribute values by some less precise ones. This can be achieved by partitioning the domain of values according to different granular scales. Let V be a domain of values for some attribute, then a *partition* π of V is a set $\{s_1, s_2, \dots, s_k\}$ of mutually disjoint subsets of V such that $\cup_{i=1}^k s_i = V$. Let π_1 and π_2 be two partitions of V . Then π_1 is a *refinement* of π_2 , written as $\pi_1 \preceq \pi_2$, if for $s \in \pi_1$ and $t \in \pi_2$, either $s \subseteq t$ or $s \cap t = \emptyset$. Also let $\pi_1 \prec \pi_2$ denote $\pi_1 \preceq \pi_2$ and $\pi_1 \neq \pi_2$. Given a number $L > 1$, an L -level domain for V is a set of partitions of V , $\Pi_L(V) = \{\pi_1, \dots, \pi_L\}$, such that $\pi_1 \prec \dots \prec \pi_L$ and $\pi_1 = V$. For each $1 \leq i \leq L$, the partition π_i is called the *i-th level partition* of $\Pi_L(V)$.

To do the level-based generalization, we have to specify the level of generalization for each attribute. Suppose that for each attribute $1 \leq j \leq m_1$, there is

an L_j such that the L_j -level domain $\Pi_{L_j}(V_j)$ for V_j is given in advance. Then a *level-based generalization* operation τ is specified by an m_1 -tuple of natural numbers $(k_1, k_2, \dots, k_{m_1})$ such that $1 \leq k_j \leq L_j$ for each j . The result of applying the operation τ to a data matrix \mathbf{T} , denoted by $\tau(\mathbf{T})$, is to replace each element t_{ij} by the equivalence class $\tau(t_{ij})$ in the k_j -th level partition of $\Pi_{L_j}(V_j)$ that contains t_{ij} . Analogously, we can also apply the operation τ to a row vector \mathbf{t} and obtain the result $\tau(\mathbf{t})$. Let τ_1 and τ_2 be two level-based generalization operations. Then τ_1 is at least as specific as τ_2 , denoted by $\tau_1 \succeq \tau_2$, if for all i, j , $\tau_1(t_{ij}) \subseteq \tau_2(t_{ij})$. Obviously, $\tau_1 \succeq \tau_2$ iff $\tau_1 \leq \tau_2$ pointwisely.

In the level-based generalization case, all operations allowed to modify the data matrix are m_1 -tuples of this kind. This answers the first question posed in the last section. To answer the remaining two questions, we need a logical language to specify the user's knowledge.

2.2 The Logical Model

To formally analyze the notion of knowledge, the so-called epistemic logic is commonly used in philosophy and artificial intelligence [Hin62, FHMV96]. Here, we will use a simplified version of the logic for modeling the user's knowledge.

Definition 2 (Simple epistemic logic) *Let \mathcal{P} be a set of atomic sentences, then*

1. *the set of objective sentences \mathcal{L}_0 is the smallest set containing \mathcal{P} and closed on the Boolean connectives \neg, \wedge and \vee ,*
2. *the set of epistemic sentences $\mathcal{L}_e = \{K\varphi \mid \varphi \in \mathcal{L}_0\}$,*
3. *the set of well-formed formulas (wffs) of simple epistemic logic is $\mathcal{L} = \mathcal{L}_0 \cup \mathcal{L}_e$.*

The objective sentences will be used to describing the property of an individual according to his record in the data table, whereas an epistemic sentence $K\varphi$ means that the user knows an individual has property φ . Thus K is a modal operator denoting the knowledge of the user. The simple epistemic language is formally interpreted on partial Kripke models [JT96].

Definition 3

1. *A partial Kripke model (possible world model) for simple epistemic logic is a triplet $M = \langle W, R, \nu \rangle$, where*
 - *W is a set of (partial) possible worlds,*
 - *R is an equivalence relation on W , and*
 - *$\nu : W \times \mathcal{P} \rightarrow \{0, 1, *\}$ is a three-valued truth assignment.*
2. *For each $w \in W$, let $\nu(w)$ be the restriction of ν to w . Then a completion of $\nu(w)$ is a classical logic interpretation $\mu : \mathcal{P} \rightarrow \{0, 1\}$ such that $\mu(p) = \nu(w, p)$ if $\nu(w, p) \neq *$ for any $p \in \mathcal{P}$. We write $\mu \sqsupseteq \nu(w)$ if μ is a completion of $\nu(w)$. Furthermore, we assume \mathcal{C} is a class of classical logic interpretations.*
3. *The satisfaction relation between classical interpretations and objective sentences is standard and denoted by $\mu \models \varphi$.*

4. The satisfaction relation between possible worlds and wffs are defined by the following clauses for $\varphi \in \mathcal{L}_0$,
- a) $w \models_M^C \varphi$ iff for all $\mu \supseteq \nu(w)$ and $\mu \in \mathcal{C}$, $\mu \models \varphi$
 - b) $w \models_M^C K\varphi$ iff for all $(w, u) \in R$, $u \models_M^C \varphi$

Note that unlike in [JT96], we do not evaluate the objective sentences in a partial possible world by a truth-functional way. Instead, the evaluation is like a super-valuation introduced in [vF71]. However, we have to parameterize the satisfaction relation by a class of classical interpretations \mathcal{C} because in the following language for describing information in data tables, not all completions of a partial interpretation are reasonable. The interpretations in \mathcal{C} are also called *reasonable* interpretations.

To describe the information in the data table, we use a special instance of simple epistemic language, called information description logic. The atomic sentences of the logic are of the form (j, α) for $1 \leq j \leq m$ and $\emptyset \neq \alpha \subseteq V_j$. In the following, the possible worlds for the partial Kripke models of the logic are just the individuals. Thus the intuitive meaning of the sentence (j, α) is that the value of an individual's attribute j belongs to α .

Let τ be a level-based generalization operation. We define $M_\tau = (U, R, \nu)$ to be a partial Kripke model for the information description logic, where

- U is the set of individuals from the context,
- $(u, w) \in R$ iff $\tau(\iota_p(u)) = \tau(\iota_p(w))$,
- $\nu : U \times \mathcal{P} \rightarrow \{0, 1, *\}$ is defined by

$$\nu(u, (j, \alpha)) = \begin{cases} 1, & \text{if } \tau(t_{ij}) \subseteq \alpha, \\ 0, & \text{if } \tau(t_{ij}) \cap \alpha = \emptyset, \\ *, & \text{otherwise} \end{cases}$$

provided that $\iota(u) = \mathbf{t}_i$.

The model reflects the user's posterior knowledge after he receives the transformed data matrix $\tau(\mathbf{T})$. Since the user can only identify the individuals by ι_p , he could not distinguish the records of u and w if $\iota_p(u)$ and $\iota_p(w)$ are generalized to the same vector by τ . Thus his knowledge is determined by the equivalence relation R .

The class \mathcal{C} of reasonable interpretations consists of those μ satisfying the following two conditions:

1. for any $1 \leq j \leq m$, there exists exactly one $v \in V_j$ such that $\mu((j, \{v\})) = 1$,
2. if $\mu((j, \alpha)) = 1$ and $\alpha \subseteq \alpha'$, then $\mu((j, \alpha')) = 1$.

We are now ready for the main definitions of this section.

Definition 4

1. The user's knowledge about the individual u after the level-based generalization τ is $IK_\tau : U \rightarrow 2^{\mathcal{L}_0}$ such that

$$IK_\tau(u) = \{\varphi \mid u \models_{M_\tau}^C K\varphi\}.$$

2. The user's general knowledge after the level-based generalization τ is the subset of \mathcal{L}_0

$$GK_\tau = \{\varphi \mid \forall u \in U, u \models_{M_\tau}^C \varphi\}.$$

3. The confidential data is a function $CON : U \rightarrow 2^{\mathcal{L}_0}$. We further assume $CON(u)$ is finite for each $u \in U$.

Definition 5

1. A level-based generalization τ is safe for u if $IK_\tau(u) \cap CON(u) = \emptyset$.
2. A level-based generalization τ is safe if it is safe for all $u \in U$.
3. Let τ_1 and τ_2 be two level-based generalization operations. Then τ_1 is at least as informative as τ_2 , denoted by $\tau_1 \sqsupseteq \tau_2$, if $GK_{\tau_2} \subseteq GK_{\tau_1}$.

Example 2 Continuing with example [1](#), assume the information description logic has the atomic propositional symbol $(d, +)$ (meaning has the disease) among others and the generalization $\tau = (1)$ (i.e., we apply the first-level generalization to the public attribute height), then the partial Kripke model $M_\tau = (U, R, \nu)$, where $U = \{u_1, \dots, u_8\}$, R is an equivalence relation such that $(u_7, u) \in R$ iff $u = u_8$, and ν satisfies $\nu(u_7, (d, +)) = \nu(u_8, (d, +)) = 1$. Thus by definition [3](#), we have $u_7 \models_{M_\tau}^C K(d, +)$, so if $(d, +)$ is confidential data of u_7 , then the generalization would be not safe for u_7 . \square

Proposition 1 Let τ_1 and τ_2 be two level-based generalization operations. Then $\tau_1 \succeq \tau_2$ implies $\tau_1 \sqsupseteq \tau_2$.

Proof: Let $M_i = (U, R_i, \nu_i)$ for $i = 1, 2$ be the model corresponding to the operation τ_i . By construction, if $\tau_1 \succeq \tau_2$, then for any $p \in \mathcal{L}_0$, $\nu_1(u, p) = *$ implies $\nu_2(u, p) = *$, and if $\nu_2(u, p) \neq *$ then $\nu_1(u, p) = \nu_2(u, p)$. So any completion of $\nu_1(u)$ must be also one of $\nu_2(u)$. By Definition [34a](#), $u \models_{M_2}^C \varphi$ implies $u \models_{M_1}^C \varphi$. Hence $GK_{\tau_2} \subseteq GK_{\tau_1}$. \square

Thus the privacy protection problem is to find the most informative (i.e., the \sqsupseteq -maximal) level-based generalization operations that are safe. Since the set of level-based generalization operations is finite (in fact they form a lattice by the component-wise ordering). We can solve the problem by performing the safety test using a bottom-up search based on the definitions.

3 Logical Model for Set-Based Generalization

3.1 The Generalization Operations

We have seen how the privacy protection problem be defined in a database linking context and solved in an effective way when level-based generalization operations are applied. However, the requirement that the L_j -level domain $\Pi_{L_j}(V_j)$ for each V_j must be a set of partitions is somewhat restrictive, since the mere partitions of a domain may not match well with the data. Sometimes, this may cause over-generalization than really necessary.

Example 3 Following Example 1, assume the data center has now a 4×2 data matrix for individuals u_1, \dots, u_4 and the identification mapping is given in an obvious way.

	height	disease
u_1	160	+
u_2	165	+
u_3	170	−
u_4	175	−

Suppose the domain of the first attribute is generalized to 4 levels with the interval length being respectively 0, 5, 10, 15 with the first interval always starts with 0 in each level. Obviously, if the goal is to prevent the user from knowing any individual's test result, then the only way is to find an interval cover all four individuals' heights. So even the coarsest level is adopted, the data table cannot be made safe. However, if we allow a bit more flexibility, then we can replace them with the imprecise values as follows.

	height	disease
u_1	[160,170]	+
u_2	[165,175]	+
u_3	[160,170]	−
u_4	[165,175]	−

Obviously, the table is safe. However, since the generalized values are not drawn from a partition, the intervals may have some overlap. \square

In the following definition, we assume a set of generalized values $\mathcal{Z}_j \subseteq 2^{V_j}$ is given for each $1 \leq j \leq m$.

Definition 6

1. A primitive operation on a data matrix \mathbf{T} is a triplet (i, j, α) , where $1 \leq i \leq n$, $1 \leq j \leq m$, and $t_{ij} \in \alpha \in \mathcal{Z}_j$.
2. A set-based generalization operation is of the form $\cap_{i \in I, j \in J} (i, j, \alpha_{ij})$ for some $I \subseteq \{1, \dots, n\}$ and $J \subseteq \{1, \dots, m\}$.
3. A row deletion operation $Rdel(i)$ is an abbreviation of $\bigcap_{1 \leq j \leq m} (i, j, V_j)$.
4. A column deletion operation $Cdel(j)$ is an abbreviation of $\bigcap_{1 \leq i \leq n} (i, j, V_j)$.

The application of a primitive operation (i, j, α) to \mathbf{T} results in the replacement of t_{ij} by α and a *set-based generalization operation* is the simultaneous application of some primitive operations. The set-based generalization operations are more flexible than the level-based ones since without predefined levels of generalization, all subsets of V_j (except the void one) can serve as the generalized values of attribute j . Unlike in the level-based case, not all records are generalized at the same level. Thus for sparse data, we may need a coarser level of generalization whereas for dense data, we can use a finer level of values. Moreover, in the set-based case, the generalization of confidential attributes is also allowed, though it is less commonly used than that for public attributes in practice.

3.2 The Logical Model

The language for modeling the user's knowledge and the class \mathcal{C} of reasonable interpretations keep unchanged as in the Section 2. Note that the generalized values are not restricted to equivalence classes in the partitions of the domain. Hence, in the set-based generalization process, it is possible that there are more than one generalized values in the transformed matrix covering the original attribute value of some individual. Thus the definition of equivalence relation R in M_τ of the last section is not applicable any more and the semantics for information description logic must be modified accordingly.

In the partial Kripke models for information description logic, both the objective and epistemic sentences are evaluated in the individuals. However, there are some differences between the interpretations of these two kinds of sentences. When an objective sentence φ is satisfied by an individual, this means that he has indeed the property described by φ *in reality*. However, if an epistemic sentence $K\varphi$ is evaluated to true in an individual, this means that the user knows the individual has the property. Since the user does not know all things about the individuals, we can treat the two kinds of sentences separately.

Definition 7

1. A double frame model for simple epistemic logic is a quadruple $M = \langle U, I, R, \nu \rangle$, where
 - U is the set of individuals,
 - I is a set of indices,
 - $R \subseteq U \times I$ is a binary relation between U and I
 - $\nu : I \times \mathcal{P} \rightarrow \{0, 1, *\}$ is a three-valued truth assignment.
2. The satisfaction relation between indices and objective wffs are defined by the following clauses

$$i \models_M^C \varphi \text{ iff for all } \mu \sqsupseteq \nu(i) \text{ and } \mu \in \mathcal{C}, \mu \models \varphi$$

3. The satisfaction relation between individuals and epistemic sentences are defined by the following clauses

$$u \models_M^C K\varphi \text{ iff for all } (u, i) \in R, i \models_M^C \varphi$$

The indices are intended to model the rows of the data matrix, so the objective sentences will be evaluated according to the individuals' records. The relation R is determined by the compatibility between the actual public attribute values of the individuals and their generalization appearing in the transformed matrix. However, since each individual must have some corresponding records in the data matrix and vice versa, some links between the individuals and the records are impossible due to other competitive links. In the next definition, we simplify a model by removing all impossible links. Before presenting the definition, let us introduce some standard notations for binary relations. Let $R \subseteq U \times I$ be a binary relation between U and I , then for $u \in U$, $R(u) = \{i \in I \mid (u, i) \in R\}$ and for $W \subseteq U$, $R(W) = \bigcup_{u \in W} R(u)$. Moreover, for $u \in U$ and $i \in I$, then

the restriction of R to $(U - \{u\}) \times (I - \{i\})$, denoted by $R_{\bar{u} \wedge \bar{i}}$, is equal to $R - ((\{u\} \times I) \cup (U \times \{i\}))$. A *perfect matching* for R is a bijection (1-1 onto mapping) $f : U \rightarrow I$ such that for all $u \in U$ and $i \in I$, if $f(u) = i$, then $(u, i) \in R$.

Definition 8 Let $M = \langle U, I, R, \nu \rangle$ be a double frame model, then the reduced model of M , denoted by M^\downarrow , is $\langle U, I, R^\downarrow, \nu \rangle$, where R^\downarrow is constructively defined by the procedure REDUCE in Figure 1.

Procedure REDUCE

Input: a binary relation $R \subseteq U \times I$.

Output: a reduced relation $R^\downarrow \subseteq U \times I$.

Begin

loop

for nonempty $W \subseteq U$ **do**

L1: **if** $|R(W)| < |W|$ **then** $R := \emptyset$; **exitloop** **endif** ;inconsistency detected

L2: **if** $|R(W)| = |W|$ and $\exists u \notin W, d \in R(W)$ such that $(u, d) \in R$
then $R := R - \{(u, d)\}$;remove impossible arcs
endif

endfor

until no more arcs can be removed

$R^\downarrow := R$

End

Fig. 1. The procedure to compute R^\downarrow

We will show that all impossible links have been indeed removed in a reduced relation in some precise sense. To show the result, the well-known Hall's theorem in graph theory is used [BM76]. A binary relation $R \subseteq U \times I$ is said to satisfy the Hall's property if for any $W \subseteq U$, $|W| \leq |R(W)|$. The Hall's theorem is as follows.

Theorem 1 (Hall) *A binary relation $R \subseteq U \times I$ satisfies the Hall's property iff there exists an injection (1-1 mapping) $f : U \rightarrow I$ such that for all $u \in U$ and $i \in I$, if $f(u) = i$, then $(u, i) \in R$.*

The injection f in the last theorem is also called a *matching saturating U* in graph theory. When $|U| = |I|$, the theorem tell us that there is a perfect matching for R if it satisfies the Hall's property. Our first result shows that if R satisfies Hall's property, then the reduction procedure will not destroy the property.

Proposition 2 *If $R \subseteq U \times I$ satisfies the Hall's property, so does R^\downarrow .*

In the next proposition, we assume $|U| = |I|$. Furthermore, a binary relation R between U and I is said to be *irreducible* if $R^\downarrow = R$.

Proposition 3 *Let $R \subseteq U \times I$ be an irreducible nonempty relation satisfying the Hall's property, then for any $(u, d) \in R$,*

1. $R_{\bar{u} \wedge \bar{d}}$ satisfies the Hall's property, and
2. there exists a perfect matching f such that $f(u) = d$

This proposition justifies our following definition for knowledge in the reduced models. Since each link from u is possible in a reduced model, all the user can know about the individual u is those true in all indices linked to u .

Let $\mathbf{s} = (s_1, \dots, s_{m_1})$ and $\mathbf{t} = (t_1, \dots, t_{m_1})$ be two row vectors, we let $\mathbf{s} \in \mathbf{t}$ if $s_i \in t_i$ for all $1 \leq i \leq m_1$. To model the user's knowledge after a set-based generalization operation τ , we can construct the following $M_\tau = (U, I, R, \nu)$ for the information description logic, where

- U is the set of individuals from the context,
- $I = \{1, \dots, n\}$, recalling that n is the row number of \mathbf{T}
- $(u, i) \in R$ iff $\iota_p(u) \in p(\tau(\mathbf{t}_i))$,
- $\nu : I \times \mathcal{P} \rightarrow \{0, 1, *\}$ is defined by

$$\nu(i, (j, \alpha)) = \begin{cases} 1, & \text{if } \tau(t_{ij}) \subseteq \alpha, \\ 0, & \text{if } \tau(t_{ij}) \cap \alpha = \emptyset, \\ *, & \text{otherwise} \end{cases}$$

Note that by the definition of R , it satisfies the Hall's property since for each u , we have an unique i such that $\iota(u) = \mathbf{t}_i$ and so $(u, i) \in R$. The definitions of user's knowledge is a slight modification of those in Section 2.

Definition 9

1. The user's knowledge about the individual u after the set-based generalization τ is $IK_\tau : U \rightarrow 2^{\mathcal{L}_0}$ such that

$$IK_\tau(u) = \{\varphi \mid u \models_{M_\tau^\perp}^c K\varphi\}.$$

2. The user's general knowledge after the set-based generalization τ is the subset of \mathcal{L}_0

$$GK_\tau = \{\varphi \mid \forall 1 \leq i \leq n, i \models_{M_\tau^\perp}^c \varphi\}.$$

The definitions of various safety properties are analogous to and follows directly from Definition 5. Thus we can effectively test whether a set-based generalization operation is safe.

3.3 On the Efficiency of Search

Though we still have an effective approach to test the safety of a set-based generalization operation according to the definition, the problem of how to find the most informative ones become more difficult in the set-based case. The degree of difficulty depends on the set of allowable generalized values \mathcal{Z}_j , so the choice of an appropriate \mathcal{Z}_j for each attribute j is a crucial factor for the efficiency of

the search problem. The most flexible way is to let the data manager specify the sets since he knows most about the data so that it is possible for him to find the most appropriate sets of generalized values according to the nature of the data tables. However, sometimes, this is considered too burdensome for the data manager, so it may be also desirable to have some default choices.

The most straightforward choice is $\mathcal{Z}_j = 2^{V_j} - \{\emptyset\}$. In this case, all subsets of V_j can serve the generalized values of the attribute j , so the data manager does not have to make any specification. However, a drawback of the choice is the easy explosion of the search space. In particular, if for some attribute j , the domain of values V_j is very large or even infinite, then the search through all subsets of V_j will become impossible. Thus, some heuristic methods may have to be employed to improve the search efficiency. For example, if the domain of values is endowed with a distance metric function $d: V_j \times V_j \rightarrow \mathcal{R}$ which satisfies some general criteria such as symmetry($d(x, y) = d(y, x)$), reflexivity($d(x, x) = 0$), and triangle inequality($d(x, y) + d(y, z) \geq d(x, z)$), then for any real number $r \in \mathcal{R}$ we can define the r -neighborhood of a value $v \in V_j$, as $N(v, r) = \{x \in V_j \mid d(v, x) \leq r\}$. The set can be seen as the r -radius sphere centering around v . Let us denote the sets of all r -radius spheres as $\mathcal{N}_r = \{N(v, r) \mid v \in V_j\}$, then the data manager can specify a number r_0 as the radius of searching. The search will start with the subsets in $\bigcup_{S \in \mathcal{N}_{r_0}} 2^S$ and continue with increasing r until the appropriate generalization satisfying the safety criteria is found.

Though the search method described in the last paragraph is more systematic than the exhaustive one, it does not prune too much of the search space since in the worst case, we have to search through the whole space if we allow any subsets of V_j as the generalized values. However, not all subsets of V_j are natural generalized values according to the semantics of the data. For example, if the attribute is height, then it does not make much sense to consider the union of two separated interval $[150, 160] \cup [170, 180]$ as a natural generalization of 175. Thus the notion of neighborhood can be used to provide a set of natural generalized values. In this case, we set $\mathcal{Z}_j = \bigcup_{r \geq 0} \mathcal{N}_r$.

There are in general two main types of attributes in data tables. The first is the nominal one and the second the numerical one. The nominal attributes seldom have a natural metric in their domains. For example, the blood type of a patient has the domain $\{A, B, AB, O\}$ which does not possess a natural distance metric. Fortunately, they in general have a smaller domain, so we can take $\mathcal{Z}_j = 2^{V_j} - \{\emptyset\}$ if j is a nominal attribute. On the other hand, it is easy to define a metric in the domains of the numerical attributes though the domains are in general large. Thus we can take $\mathcal{Z}_j = \bigcup_{r \geq 0} \mathcal{N}_r$ if j is a numerical attribute. Sometimes, though a natural metric exists for the domain of an attribute, not all r -radius neighborhoods in \mathcal{N}_r provide natural generalized values for the attribute. For example, if the birth date of a patient is "2000, Dec., 25", then it is natural to generalize the data to "2000, Dec." or "2000". However, it definitely does not make much sense to generalized it to "the interval from 2000, Dec., 15 to 2001, Jan., 15" though it has the same length as "2000, Dec.". For this kind

of attributes, we can just set \mathcal{Z}_j as the L_j -level domain $\Pi_{L_j}(V_j)$ introduced in the last section.

4 Conclusion

In this paper, we have presented a logical model for privacy protection in the database linking context. To avoid privacy violation, the data table may have to be modified before they can be released to the user. A simple epistemic logic and one of its special instances, called information description logic, are employed to model the user's knowledge. According to the model, a safety criterion of the data is defined and can be tested effectively. Moreover, we also discuss the approach of finding the most informative data satisfying the safety requirement.

The safety criterion defined in this paper can be seen as a generalization of bin size [Swe97]. According to the bin size criterion, the individuals having a particular public attribute value must be non-unique. Here we require not only the non-uniqueness but also the non-uniformity conditions. That is, the individuals having a common public attribute value must have different confidential attribute values. This criterion is purely qualitative. We can further consider some quantitative criteria of safety. This will rely on the distributions on the different values of the confidential attributes. In this regard, we may have to extend the epistemic logic with some probabilistic reasoning mechanism [FH94, Hal98].

References

- [BM76] J.A. Bondy and U.S.R. Murty. *Graph Theory with Applications*. Macmillan, New York, 1976.
- [Cam00] L.J. Camp. *Trust and Risk in Internet Commerce*. The MIT Press, 2000.
- [Chi00] Y.C. Chiang. Protecting privacy in public database (in Chinese). Master's thesis, Graduate Institute of Information Management, National Taiwan University, 2000.
- [CHKW00] Y.-c. Chiang, T.-s. Hsu, S. Kuo, and D.-w. Wang. Preserving confidentiality when sharing medical data. In *Proceedings of Asia Pacific Medical Informatics Conference*, 2000.
- [Den82] D.E.R. Denning. *Cryptography and Data Security*. Addison-Wesley Publishing Company, 1982.
- [FH94] R. Fagin and J. Halpern. "Reasoning about knowledge and probability". *Journal of the ACM*, 41(2):340–367, 1994.
- [FHMV96] R. Fagin, J.Y. Halpern, Y. Moses, and M.Y. Vardi. *Reasoning about Knowledge*. MIT Press, 1996.
- [Hal98] J. Halpern. "A logical approach to reasoning about uncertainty: a tutorial". In X. Arrazola, K. Korta, and F.J. Pelletier, editors, *Discourse, Interaction, and Communication*, pages 141–155. Kluwer Academic Publishers, 1998.
- [Hin62] J. Hintikka. *Knowledge and Belief*. Cornell University Press, 1962.

- [JT96] J. Jaspars and E. Thijsse. “Fundamentals of partial modal logic”. In P. Doherty, editor, *Partiality, Modality, and Nonmonotonicity*, pages 111–141. CSLI Publications, 1996.
- [Mor88] M. Morgenstern. “Controlling logical inference in multilevel database systems”. In *Proc. of the IEEE Symposium on Security and Privacy*, pages 245–255, 1988.
- [Paw91] Z. Pawlak. *Rough Sets—Theoretical Aspects of Reasoning about Data*. Kluwer Academic Publishers, 1991.
- [Swe97] L. Sweeney. “Guaranteeing anonymity when sharing medical data, the Datafly system”. A.I. Working Paper AIWP-WP344, MIT AI Lab., 1997.
- [vF71] B.C. van Fraassen. *Formal Semantics and Logic*. Macmillan, New York, 1971.

DISSECT: DIStribution for SECurity Tool

Enriquillo Valdez^{1*} and Moti Yung²

¹ Polytechnic University, Computer and Information Science Department
Brooklyn, NY, USA

`rvaldez@photon.poly.edu`

² CertCo Inc., New York, NY, USA

`moti@cs.columbia.edu`

Abstract. A security threat that affects the Java environment (as a typical environment where code is made available to the user machine) is the reverse-engineering and code-understanding of the architecture-neutral bytecode format. A natural protection strategy is to hide parts of the execution in trusted locations (e.g., servers). However, the implementation and automatization of such an approach (beyond the abstract idea) is a challenging problem. In this paper, we present a novel software protection strategy and its automatization (implemented architecture) which materialize the above idea. It is employed in protecting the binary source of Java class files. Our software protection strategy partitions “programmer selected” classes of an application into server classes and client classes. Server classes contain the actual class code and run only on trusted systems (which we call servers but they can be other dedicated machines). Client classes, on the other hand, are assumed to perform most of the task (but the sensitive part) and execute on user systems; they must interact with their corresponding server class in order to execute the sensitive code and provide the behavior of the original class. We propose and implement DISSECT (DIStribution for SECurity Tool), an architecture based on the above partitioning (dissection) strategy, for Java 1.1. The tool relieves the developers from actually writing distributed applications by distributing the application automatically, according to designated sensitivities of application portions. We note that the remote execution of classes may increase the overhead. Thus, we have conducted initial experiments to understand the impact of partitioned classes on performance. We report initial performance results which show the overhead and demonstrate when it is low or non-existing, when it is high, and when we actually gain performance by partitioning.

1 Introduction

Java has emerged as the technology of choice for harnessing the nascent network computing environment. In such an environment, applications need to be

* This work was supported by the National Science Foundation under Grant No. 9256688 and the NY State Center for Advanced Technology in Telecommunications. The author is now with IBM T. J. Watson Research Center, Hawthorne, NY. E-mail: `rvaldez@us.ibm.com`.

lightweight, flexible, and extendable. Applications download their software components as needed from the network at runtime. With built-in networking support, dynamic loading capability, and platform independent code, Java is suitable for such an environment. However, supporting dynamic loading requires a significant amount of symbolic information (e.g., method names, signatures, access specifiers, etc.) to be maintained in the Java class files. To provide platform independence, the Java runtime system (i.e., Java Virtual Machine) is configured to operate in an environment that represents the lowest common denominator in all hardware architectures, a stack machine. Unfortunately, these characteristics make Java code very easy to decompile [11,15,29,33], and thus software developers are apprehensive about producing their software products in Java, because they fear that their private knowledge components (e.g., trade secrets, proprietary algorithms, etc.) will be reverse-engineered.

In this paper, we present the design and implementation of a network oriented protection architecture to prevent the reverse-engineering of Java applications. The basic idea behind this architecture is to automatically and systematically remove valuable components from the application code delivered to users and to provide access to these distributed components via network resources. While removing parts of execution may be a natural ad-hoc idea, the automatization of it into a tool which performs the splitting of an application is the challenging issue and a powerful idea which we attempt to cope with.

In our methodology and the tool (DISSECT) implementing it, selected classes of an application are partitioned into client (local) and server (remote) classes. Consisting mostly of remote method calls, a client class encapsulates the visible behavior of the original class from a user's perspective (e.g., provides access to public methods and variables). The server class contains the original class code extended to support requests from the client class. A client object (i.e., an instantiation of a client class) must interact with its corresponding server object to provide the behavior of the original object. The client classes are distributed to and executed by users, and the server classes are delivered to trusted systems. Our architecture relies on Remote Method Invocation (RMI) [5] to enable communication between client and server applications.

Our approach is applicable to situations where software developers want to prevent the disclosure of embedded secrets in programs or inner workings of programs. Suppose a program encrypts some data using an embedded cryptographic key. Applying our approach, the partitioned program maintains the cryptographic key on trusted servers and not embedded in the user's program (this is, by the way, in line with common practice regarding cryptographic co-processors [27]). Thus, to generate encrypted data, a user interacts with a server that receives the user data and returns the encrypted data. Our tool will automatically transform a program where the encryption and other cryptographic management operation are marked as sensitive to a distributed program where the entire cryptographic portions are performed at the server. Another application of this idea is a mobile agent that executes on an untrusted host [31] but performs all its cryptographic or proprietary operations remotely on a trusted

server; again our tool will be used to transform the agent into a distributed program executing the sensitive part remotely.

Though we can hide code and data with DISSECT, we cannot hide the input/output relationship of partitioned classes because we allow information to flow back to the application on the client side. However, this provides a much weaker understanding of the code than divulging the code itself.

Selectivity needs to be applied when adopting DISSECT methodology, since it may not be suitable for all types of applications (e.g., non-networked or embedded applications). Partitioned applications will incur a communication penalty due to invoking methods on remote systems. For programs that already access an external resource, such as a database on the network, we may want to partition according to such external accesses. Again, the program can be written as though it is running at the same location, and the tool is then used to perform the required partitioning according to the designated sensitivities.

In general, we should try to compensate for added overhead communication by executing server objects (i.e., sensitive program portions) on systems that offer advantages in terms of computing resources, such as a high-performance processor with cryptographic hardware. Selecting classes for partitioning will require developers' discretion and foresight, but is much less work than performing the partition at the development stage in an ad-hoc fashion (rather than employing our tool). In the ideal situation, a partitioned application should spend most of its time executing either on the client or the server site and not waiting for remote method returns.

This paper is organized as follows: Section 2 discusses related software protection and partitioning work for the Java environment. Section 3 describes in general the DISSECT methodology for object-oriented applications. In Section 4 we present some details on DISSECT architecture and the implementation for the Java environment. Section 5 reports initial performance results on partitioned applications. Section 6 discusses the programming style and assumptions that allow the use of DISSECT. Section 7 contains the conclusions.

2 Related Work

Technical approaches to address reverse-engineering for Java have included obfuscating classes, computing with encrypted functions, and executing objects inside a tamper proof environment.

Obfuscation techniques [8,9,16,28] have focused on manipulating portions of the programs. These techniques have added extraneous symbols, removed unnecessary symbols, or replaced symbols with confusing or illegal strings in the constant pool, the data structure that maintains symbolic and type information in the Java class binary. Collberg et al. [6,7] describe obfuscation techniques based on compiler optimization techniques. In their work, they introduce the notion of opaque constructs which are variables or predicates whose values (or properties) are difficult for deobfuscators to determine. These opaque constructs are used to prevent the removal of extraneous code that they add to Java pro-

grams. Obfuscation only serves to make it difficult to reverse-engineer. Theoretical implications of obfuscation and its impossibility are discussed in [11,3].

Another approach to protection, based on hiding programs uses cryptographic primitives. Sander and Tschudin [20] employ encrypted functions to hide function evaluations by mobile agents. Unfortunately, their theoretically appealing method applies only to very specific polynomial functions (this work was extended by Sander et al. [21]). Another approach links execution of classes with tamper resistance hardware. Wilhelm [30] presents an environment for executing objects that prevents disclosure (and tampering) of their inner workings. The classes are delivered to users encrypted; the classes are then decrypted and executed only within a tamper proof device, which contains the decrypting keys. This approach requires special hardware installed on user systems and is not portable.

Practical work relevant to our Java partitioning methodology has also appeared. In terms of preprocessing Java code to execute on remote systems, Malkhi et al. [17] address malicious code threat for applets (the dual problem to ours!). In their work, instead of running within a user's browser, a downloaded applet is processed to execute in a sanitized environment where it cannot harm the user's system. The processed applet has its input/output (I/O) redirected to the web browser (that downloaded it). Our work differs from theirs in the following aspects: (i) in our work, users never get the actual bytecode of a protected class, but only a skeleton of the class. The processing of the Java code is done by the software developer instead of the end user as in [17]. (ii) Our approach is fine-grained where an application is partitioned on the class level to execute on different remote systems. In [17], a coarse-grained approach is presented where the entire applet executes on a remote system, having its I/O redirected.

In a seemingly related concurrent work, Balfanz [2] addresses the problem of collaboration between a trusted hand-held device that is resource constrained and a powerful untrusted computer. His splitting-trust framework enables an application that runs in a hand-held device to execute parts which are not sensitive (e.g., the user interface) on an untrusted computer. He partitions by simply instantiating classes of an application on the untrusted computer, but in an ad-hoc fashion and not by using an automated tool. Developers must explicitly program their Java applications to make use of his Java splitting-trust API. In contrast, DISSECT automatically and systematically partitions at the class element level by removing elements from classes that are deemed sensitive.

3 DISSECT Overview

In this section, we present an abstract description of the DISSECT methodology devoid of implementation details. The methodology consists of three aspects. First, we select the code to be partitioned. Second, we process the selected code. Third, we organize the distribution and execution of the partitioned code. The methodology is designed to operate on any object-oriented programming lan-

guage that supports remote method invocation. Next, we present some relevant information on object-oriented systems.

3.1 Object-Oriented Systems

In an object-oriented system, an *application* (or a program) consists of a collection of classes. A *class* defines a set of elements consisting of variables (i.e., fields) and methods (i.e., functions). An *object* is an instantiation of a class. The object's *behavior* is defined by its methods, and the object's *state* by its variables. With class elements, we associate an access attribute and a modifier attribute. The access attribute specifies the visibility of the class element. To simplify the discussion, this section only considers *private* and *public* access attributes. Private class elements are only accessible to the class itself (no external reference), and public class elements are accessible to all classes. The modifier attribute indicates whether a class element is a static element or an instance element. A static class element is specific to the class; an instance element is associated with an instance of a class.

An object-oriented system also supports the notion of *inheritance*, which allows a class (*subclass*) to share the behavior of another class (*superclass*), in addition to specifying its own. In this paper, we consider object-oriented systems that supports single inheritance hierarchy such as Java.

3.2 Selection Criteria

In the DISSECT methodology, some classes of an application are selected as sensitive and we wish to prevent reverse-engineering of these classes. We can select a class for partitioning based on the following set of rules:

1. The class contains something of value.
2. The performance of the class (or its instantiation) is not penalized greatly by executing it remotely.
3. The class is not closely coupled with the client (user) system.

The first rule states that a class is a good candidate for partitioning if it is valuable. We consider a class to be valuable if it implements a complex piece of code, or if it contains embedded secrets that a developer wants to protect from inspection or reverse-engineering. Examples of embedded secrets in programs are proprietary algorithms or cryptographic keys. The second rule considers a class for partitioning if its remote execution (or its remote instantiation) does not overly penalize the performance of the application. In some cases, remote servers, such as a high-performance processor with cryptographic hardware, can offer a computational advantage over the user systems in terms of computing resources. The third rule recognizes that if a class (or its instantiation) depends strongly on some locally available resource then it is not a good candidate for partitioning. In the Java environment, an example of such a class is the `java.awt.Graphics` class, which interacts with platform dependent code. Note that the first and

second rules, security and efficiency, are in conflict. In general, the second rule may not always be satisfied because of the communication cost associated with partitioned classes (i.e., between client and server objects).

3.3 Processing

Selected classes are processed by applying a partitioning strategy. The partitioning strategy operates on a class as follows. Each selected class is partitioned into a client (local) class and server (remote) class. A class element with private access appears only in the server class. A class element with public and static attributes appears in both client and server classes. An instance class element does not appear in the client class. For an instance method, the client class has a proxy method that invokes the corresponding method on its server object. For an instance variable, the client class has access methods for setting and retrieving the variable from its server object. In contrast, the server class contains the actual instance methods corresponding to the client class's proxy methods and the access methods for variables. Note that all the superclasses in the inheritance tree of a selected class are partitioned as well. In Section 4.1, we discuss the processing step for Java classes in detail.

3.4 Organizing Distribution and Execution

When we apply the DISSECT tool to an application, client and server classes are generated from the selected classes. When the application is distributed to users, it has selected classes replaced with the generated client classes. We call this application a *client application* and its corresponding set of server classes a *server application*. We call the system where a client application resides a *client system* and remote systems where server applications are instantiated *server systems*. Thus, to provide the functionality of the original application, the client application must interact with its corresponding server application over some network. Note that this breaks from the traditional Java paradigm of server systems providing code and client systems executing code. In our architecture, both server and client systems are involved in executing a partitioned application.

With respect to the runtime environment, we make the following assumptions: both client and server systems have access to a network and that connections over the network are reliable and secure. In addition, we assume server systems to be trusted, available, and willing to participate in client application executions. We observe that if we apply the DISSECT tool to an application and execute the partitioned application in this environment, we prevent the reverse-engineering and code-understanding of the selected classes. This is so since users do not get access to the selected classes but only to the proxy classes.

4 DISSECT: Architecture and Implementation

This section presents some implementation details on the DISSECT methodology for the Java environment.

The DISSECT architecture consists of a processing stage and an initialization stage. In the processing stage, an application is transformed into a client application and a server application. This is done only once. In the initialization stage, objects of the client application link with their respective server objects. This is always done at runtime.

4.1 Processing

In the processing stage, the DISSECT tool performs several tasks. First, it partitions selected Java classes into server and client classes. Second, the tool generates other classes to support the execution of the partitioned application; it generates an Application Class Loader, a global class, and a new application main class. The Application Class Loader is responsible for instantiating server classes on server systems. The global class maintains essential information (e.g., name of the Application Class Loader, etc.) to facilitate the instantiation of client classes from anywhere in the application's class tree. In addition to the functionality of the original main class, the new application main class instantiates the **RMISecurityManager** class (this is necessary when interacting with remote classes) and retrieves server system addresses from network directories (e.g., Server Name Services). Finally, the tool adapts the application to work with partitioned classes: references to classes that are now partitioned are updated to reflect the appropriate classes (i.e., client or server class) in unpartitioned classes. The inputs to this stage are the Java class files of an application and a list of selected classes to partition; Figure 4.1 shows the class files generated.

In addition to the processing stage, we implemented Remote Host Objects (*RHOS*) (i.e., customized server IO classes) in order to provide input/output capabilities of a client system to a sever system. In the standard Java class library, there are API classes that implement neither the `java.io.Serializable` interface nor the `java.rmi.Remote` interface; this means that these classes are not serializable or accessible remotely. However, in certain situations, having access to such API classes is appropriate for server classes; the original (unprocessed) class may contain such a reference (e.g., a call to the `java.System.out.println()` method).

In this subsection, we only discuss the partition strategy for Java classes due to the extensive detail associated with the other components of this stage. In addition, we provide some low level detail on the Java language interface to the RMI mechanism to enable understanding of features of partitioned classes. We also present an example of a partitioned class and discuss the assumptions we make on the use of DISSECT in the Java environment.

The Java Partitioning Strategy. The Java partitioning strategy works by extracting and encapsulating the behavior and state of an object which is visible to the user. As discussed in Section 3.3, the strategy partitions an original class into client and server classes.

In the Java environment, static class elements have the keyword `static` in their declarations and instance class elements do not. The Java partitioning strategy

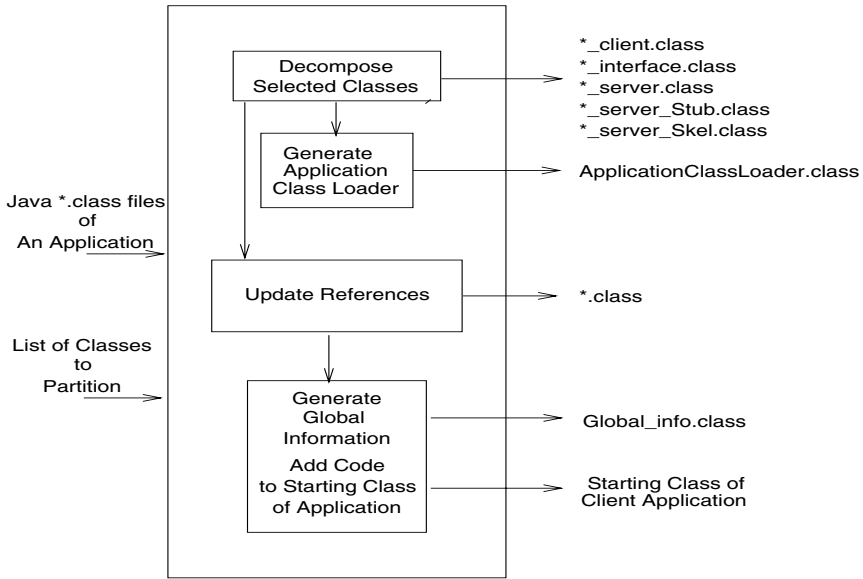


Fig. 1. The processing stage. The `*_server.class` and the Application Class Loader class files are distributed only to server systems. The `*_client.class` and other class files are distributed to client (user) systems. The `*_interface.class` class files are distributed to client and server systems.

reproduces static class elements on both the client and server classes. In the Java environment, there are also four types of access attributes: `public` (if the `public` keyword is specified), `protected` (if the `protected` keyword is specified), `private` (if the `private` keyword is specified), and `package` (if no keyword is specified). The accessibility of `public` and `private` class elements is described in Section 3.1. A class element declared as `package` or `protected` is visible to all the classes of the same package namespace. In addition, a class element declared as `protected` is also visible to the subclass regardless of the package namespace. For `protected` class elements, we assume that only the subclasses have access to these elements in the applications. Because of this, `protected` instance elements need only appear in the server class. Thus, the partitioning strategy converts `protected` to `private` access. It also converts `package` to `public` access. This is done to provide access of package class elements of a partitioned class to unpartitioned classes within the same package namespace. In summary, the Java partitioning strategy processes an instance class element as follows: a class element with `private` or `protected` access is processed to appear in the server class; a class element with `public` or `package` access is provided by having a proxy or an access method in the client class, while the actual class element appears on the server class. A client class, for example, has proxy methods and access methods for setting and retrieving the variables from the server object.

In the Java partitioning strategy, once a class is selected for partitioning, all its superclasses are automatically selected as well. Note that only the top-level class of the hierarchy makes the request for a server class instantiation on the remote system. A top-level class is a subclass of the `java.lang.Object` class.

To provide the instance behavior and state of the original class, a client object must interact with its corresponding server object. Therefore, the partitioning strategy enables a client object to request its corresponding server class from a remote system at instantiation time. For each constructor method in the original class, the strategy produces a corresponding method on the client class. In the client class, the constructor methods are augmented to receive arguments (i.e., machine addresses and port numbers) for connecting with a server system.

Java RMI Mechanism. DISSECT uses Java’s RMI mechanism to enable communication between a client object and a server object. Using RMI requires a remote interface (i.e., one that extends `java.rmi.Remote` directly or indirectly) to be defined. This interface has public access and declares all the remotely available methods. (Methods declared in an interface file cannot have an access modifier of `native`, `synchronized`, or `final`. Thus, if an original class has methods with such access modifiers, DISSECT processes such methods to appear only in the server class.) Every declared method in the interface throws at least the exception `java.rmi.RemoteException`. In Section 4.2’s Failure Modes, we discuss how client objects can address this type of exception. In the DISSECT methodology, both client and server classes implement the methods of their corresponding remote interface. (Recall that the client class implementation consists mostly of remote calls to its server object.) In the Java environment, our server classes are required to be remote objects. A remote object is an object whose methods can be invoked from objects running on different Java virtual machines. A remote object must extend the `java.rmi.server.RemoteServer` class; however, this is an abstract class. The only concrete RMI server class provided with the Java Development Kit (JDK) (for version 1.1.) is the `java.rmi.server.UnicastRemoteObject` class which provides the basic underlying support for a single server object. Hence our generated server class extends the `java.rmi.server.UnicastRemoteObject` class. Other classes automatically derived from a remote class are the stub and skeleton classes. The stub and skeleton classes handle the low level communication between client and server objects. In Section 4.2, we discuss how client objects obtain references to server objects.

An Example of Partitioning. To demonstrate the Java partitioning strategy, we explain the outputs on a simple Java class, `Circle.java`, given in Figure 2(a). Though we show the source of the original Java class for illustration purposes, DISSECT partitions Java class files (i.e., Java binary files and not source files) and generates new Java class files. Figures 2(b,c,d) show actual classes generated by our partitioning tool that were subsequently decompiled by Mocha [29].

When applied onto a class, the Java partitioning strategy generates three classes: a shared interface class, a client class, and a server class. The shared

interface class (see Figure 2(b)) declares all remotely available methods of the server class. These methods correspond to instance elements declared with public or package access in the selected class. The `Circle` class has one method and two variables with public access. For these class elements, the shared interface, `Circle_face`, declares a proxy method, `calArea()`, and two access methods, `setValuePrimDouble()` and `getValuePrimDouble()`. The `setValuePrimDouble()` and `getValuePrimDouble()` methods set and retrieve instance variables, `x` and `y` (see the server class in Figure 2(d) where these methods are implemented). The `Circle_face` class also declares an initialization method, `DD()`. This initialization method corresponds to the `Circle` class's constructor method with arguments.

Both the client and server classes implement the methods declared in the shared interface. Figure 2(c) shows the client class, `Circle_client`, which is a top-level class. Except for the constructor methods, the `Circle_client` class contains simple methods that invoke their remote counterparts on the server object. The first constructor method in the class is responsible for requesting the remote instantiation of the server class on the server system. The second constructor method first calls the first constructor method (i.e., `this()`) and then the remote initialization `DD()` method (via a call to `ShapeCircle_hostRemote.DD()`¹) to pass the constructor arguments to the server object. Note that we omit RMI exception handling code from Figure 2(c).

Figure 2(d) shows the server class, `Circle_server`, for the `Circle` class. The server class implements the methods of the `Circle_face` and extends the `UnicastRemoteObject` class as required. It has the original class's methods and variables; it implements the access methods (`setValuePrimDouble()` and `getValuePrimDouble()`) for returning and setting public variables. The stub and skeleton classes are not shown, but they are also generated for the `Circle_server` class.

4.2 Initialization

This subsection presents the initialization stage and describes techniques for handling errors associated with remote method invocation. We make several assumptions regarding our implementation: first, that both client and server systems have access to some network; second, that connections over the network are reliable and secure and third, that server systems are trustworthy, available, and willing to participate in client application executions. In our implementation, we assume the existence of *Server Name Services* (SNS)s. SNSs maintain a list of addresses of available server systems. The addresses of SNSs are assumed to be known; they can be retrieved, for example, from a central web site. We also assume that server systems are running and are waiting for requests from client applications to start their corresponding server applications.

In the initialization stage, the objects of a client application establish links to their corresponding server objects on server systems. Figure 3 shows the required initial steps when a partitioned application starts running.

¹ The `DD()` method is an empty method in the `Circle_client` class. This fulfills the requirement that the client class has implemented all the methods of the `Circle_face` interface.

<pre>// FIGURE (a) package Shape; public class Circle { public double x,y; private double r = 11.0; private double Pi = 3.14; public Circle() { x = 3.0; y = 7.0; } public Circle(double x, double y) { this.x = x; this.y = y; } private double area() { return Pi * r * r; } public double calArea() { return (x * y * r * area()); } }</pre>	<pre>// FIGURE (b) package Shape; import java.rmi.Remote; import java.rmi.RemoteException; public interface Circle_face implements Remote { public abstract void DD(double d1, double d2) throws RemoteException; public abstract double calArea() throws RemoteException; public abstract void setValPrimDouble(double d, String string) throws RemoteException; public abstract double getValPrimDouble(String string) throws RemoteException; }</pre>
<pre>// FIGURE (c) package Shape; ... public synchronized class Circle_client implements Circle_face { ... // First constructor method establishes link with server object public Circle_client(...) { ... AppRemoteServ = (HostLoaderX)Naming.lookup(... "/Loader" ...); Decomp_genfaceRemoteObj = AppRemoteServ.LoadByName(...); ShapeCircle_hostRemoteObj = (Circle_face)Decomp_genfaceRemoteObj; ... } // Second constructor method public Circle_client (...) { this(...); ... // Calls initializing method on server object ShapeCircle_hostRemoteObj.DD(d1, d2); } // An empty method, not needed on client side public void DD(double d1, double d2) {} // The proxy for invoking calArea on the server object public double calArea() {.... double d = ShapeCircle_hostRemoteObj.calArea(); return d; } // Proxy access methods for getting and setting variables public double getValPrimDouble(String string) {.... double d = ShapeCircle_hostRemoteObj.getValPrimDouble(string); return d; } public void setValPrimDouble(double d, String string) {.... ShapeCircle_hostRemoteObj.setValPrimDouble(d, string); } }</pre>	<pre>// FIGURE (d) package Shape; ... public synchronized class Circle_server extends UnicastRemoteObject implements Circle_face { public double x; public double y; private double r; private double Pi; // called by the ApplicationClassLoader public Circle_server(Vector vector1, Vector vector2) { Pi = 3.14; r = 11.0; x = 3.0; y = 7.0; } public Circle_server() {} // Initialization method public void DD(double d1, double d2) { Pi = 3.14; r = 11.0; x = d1; y = d2; } private double area() { return Pi * r * r; } public double calArea() { return x * y * r * area(); } // Implementation of Access methods public double getValPrimDouble(String string) { double d; if (string.equals("x")) d = x; else if (string.equals("y")) d = y; ... return d; } public void setValPrimDouble(double d, String string) { if (string.equals("x")) x = d; if (string.equals("y")) y = d; ... return; } }</pre>

Fig. 2. (a) Shows a simple Java class, (b) shows the generated interface class, and (c) shows parts of the generated Circle client class, and (d) shows parts of the generated Circle server class (we omit exception handling code)

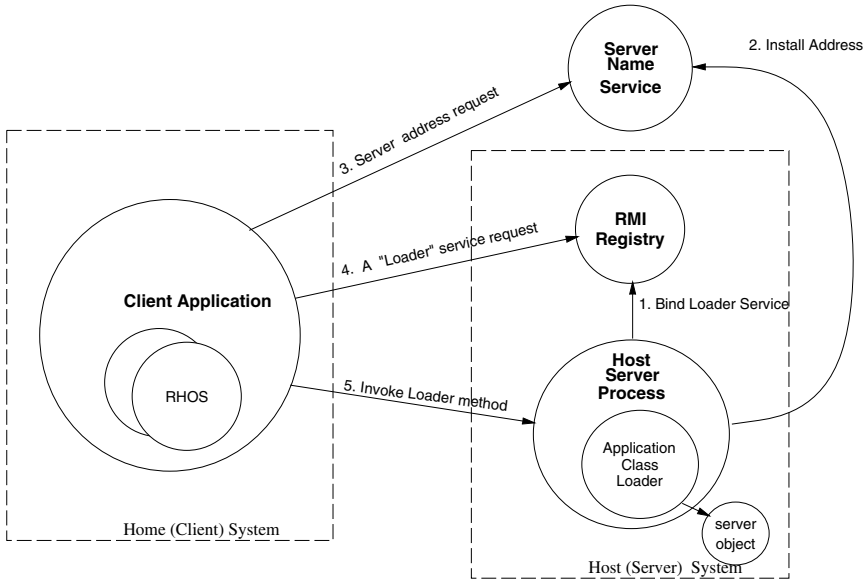


Fig. 3. Initialization Stage

A server system executes a Host Server Process (*HSP*)², which services client object requests. The HSP supplies methods that instantiate and return references of server objects running on the server system. At start up, the HSP registers a reference of itself on a locally running registry (e.g., the RMI registry) with the name *Loader* (see Figure 3 [1]). A request for the *Loader* service on the registry returns a remote reference to the HSP. The HSP then forwards its machine address and the port identification (i.e., port number) of the locally executing registry to SNSs [2].

In a client system, when a client application starts executing, it retrieves the addresses of server systems from SNSs [3]. The client application then sends requests to the registries at these locations for references to *Loader* services [4]. Within the client application, every client object invokes a *Loader* service passing the following parameters: the name of the *ApplicationClassLoader*, the name of the server class (to instantiate), and references to RHOS [5]. (In our example, the first constructor method in the *Circle_client* class invokes the *Loader* service.) Once HSP receives these parameters, it instantiates the specified *ApplicationClassLoader*. The *ApplicationClassLoader* object instantiates the requested server class. If the server class instantiation is successful, the client object gets a reference to its remote counterpart (i.e., server object). Once

² DISSECT can also employ the Jini technology infrastructure [26] to advertise the services of the HSP(s).

³ By referencing addresses of server systems cached from previous executions, applications can avoid interacting with SNSs.

the client application classes obtain the remote references to their application server classes, the execution of the application can proceed.

Failure Modes. Partitioning introduces new failure modes to applications. A partitioned application must deal with exceptions associated with remote method invocation. RMI exceptions may result from network uncertainty (e.g., lost packets), failure of server systems (e.g., crashes), server objects being prematurely collected by the garbage collector on server systems, RMI classes (such as stubs classes) not being found at runtime, etc. We provide two approaches for handling RMI exceptions caused by server objects being unavailable (e.g., when a client object is unable to link with its corresponding server object, or when client objects' references to server objects become invalid).

In the first approach, when an RMI exception occurs, the partitioned application reports the error (by writing to the Standard Error Device of the client system) and terminates. This, however, may not be reasonable for some (long running) applications. In the second approach, attempts are made to recover from the RMI exception. The partitioned application first tries to reconnect with the server system that originally provided the server object. If unsuccessful, it then tries to establish a connection with another server system. If unsuccessful after a predetermined number of tries, the partitioned application reverts to the first approach.

Implementation. We implemented a prototype of the protection architecture for Java 1.1. (When this work was done, Java 1.1 was the currently available version.) Our tool takes as input: the name of the starting class of the application, option parameters, a list of class names to partition, and the Java binary class files of the application. We used freely available software to aid in the partitioning of Java classes. Supplied in [32], classes of the Jcf package were used for reading Java class binaries into a data structure suitable for manipulation. To this package, we added classes to enable us to read the bytecode of a Java class binary. Because generating a Java class binary directly is a cumbersome process due to the indirection associated with entries in the constant pool, we used Jasmin [14], a Java assembler, to generate Java class binaries. Thus, our tool generates ASCII descriptions of Java classes, which are then assembled to (Java class) binaries by using Jasmin.

5 DISSECT: Performance

We evaluate the performance of our approach by measuring and comparing the execution time of method invocations of partitioned applications with original applications. Our experiments were conducted on Sun ULTRAs, running Solaris 2.7, connected by a 10 Mbps Ethernet LAN on the same subnet⁴. The systems

⁴ The proximity of machines may seem unrealistic; however, one can execute a server system on a secure co-processor (e.g., IBM 4758 [13]) in the client system!

designated as server systems were lightly loaded. For our experiments, we used Java 1.1.6 version 3 with the just-in-time (JIT) option set off. Obviously, we prefer to have the JIT option on, since this would speed up (possibly by a factor of 10-20) the execution of the applications. However, the JIT mechanism experiences a runtime error when executing the server classes generated by our tool⁵. We obtained explicit timing measurements by instrumenting our applications with calls to `System.currentTimeMillis()` before and after a method invocation. Table 2 shows the timing results of the methods in which a data value represents the average of 5000 method invocations except for entries with a star symbol, which were calculated from the average of 100 invocations. We evaluate the performance of four applications.

Table 1. _209.db Application

Application	Method	Description	Operations
_209_db	Database.Database()	1 arg/no result	initialize database
	Database.list()	no args/no result	show beginning record
	Database.next()	no args/no result	increment record pointer
	Database.add()	1 arg/no result	add record
	Database.sort()	2 args/no result	sort records
	Database.modify()	1 arg/no result	modify record
	Database.find()	2 args/no result	locate record
	Database.remove()	no args/no result	delete record

5.1 Applications

The first application, _209_db, is a Java database benchmark program from the Standard Performance Evaluation Corporation (SPEC)^[22]. This database applies operations on memory resident data. The operations of the database methods are listed in Table 1; each data record consists of a person’s name, an address, and an identification number. For our performance experiments, the `Database.Database()` constructor method reads a 1 MB data file containing 15532 records. The second application, CompPi, calculates pi to the precision of 50 digits. We select the Pi class which performs the Pi calculation for partitioning. The `Pi.Pi()` constructor method receives the parameter for setting up the precision, and the `Pi.execute()` method computes Pi by using Machin’s formulas^[4]. The `Pi.execute()` method returns an object of the type `java.lang.BigDecimal`. The third application, DataB, simulates the functionality of a sparse multiuser database. Here, we partition the `DataApp` class which maintains memory resident data. The `DataApp` class provides methods (`method64()`, `method32()`, and `method4()`) for retrieving records stored in memory. These methods randomly

⁵ Since our bytecode is not generated by Sun’s JDK compiler, this thwarts the operations of Sun’s JIT^[19] (p. 135).

Table 2. The average execution time of remote and local method invocations (5000 calls) (* 100 calls)

Application	Method	Remote (ms)	Local (ms)	Remote/Local
_209_db	Database.Database()* ⁶	6515.61	6250.5	1.04
	Database.list()*	117.68	101.62	1.15
	Database.next()*	17.74	9.56	1.85
	Database.add()*	18.58	0.03	619.3
	Database.sort()*	8565.8	5741.29	1.49
	Database.modify()*	8.54	0.43	19.86
	Database.find()*	5417.82	3460.11	1.56
	Database.remove()*	856.8	450.15	1.90
CompPi	Pi.Pi()*	860.19	0.21	4096
	Pi.execute()	24.58	17.09	1.43
DataB	DataApp.DataApp()*	941.07	24.14	39
	DataApp.method4()	9.05	0.082	110
	DataApp.method32()	19.73	0.449	44
	DataApp.method64()	23.06	0.872	26
MainXor	SimpleXor.SimpleXor()*	1548.1	9643.37	.160
	SimpleXor.rcvData()	3.91	1.03	3.8
	SimpleXor.retKey()	2.89	N/A	N/A

pick records to return; the return parameter of these methods is a `Vector` containing `java.lang.String` objects⁶. In the `DataApp.DataApp()` constructor method, a local data file containing the database records is read and the state of the database is initialized. In the fourth application, `MainXor`, a local text file is read and passed as data to the `SimpleXor` class which performs a simple encryption. We select the `SimpleXor` class for partitioning. The local text file resides on the client system. In `SimpleXor`'s constructor method, an instance of the `java.security.SecureRandom` class is instantiated. The `SecureRandom` object takes approximately an average of 7 seconds to generate its random seed. The `rcvData()` method (of `SimpleXor`) receives two parameters, a data array of 80 integers and the length of the array to process. This method requests 80 random bytes via a call to the `java.security.SecureRandom.nextByte()` method, applies an exclusive OR operation on the data array with the randomly generated key, and returns an encrypted version of the data array. The `retKey()` method returns the randomly generated key used in the last invocation of the `rcvData()` method.

We consider these applications to be good candidates for DISSECT. A developer, for example, may wish to hide the data and retrieval algorithms of the database applications, `_209_db` and `DataB`. Performance and hiding implementation details are the reasons for partitioning the `CompPi` and `MainXor` applications.

⁶ The number at the end of the method indicates the number of `String` objects contained in the `Vector` Object.

5.2 Results

Our results show that local method invocation is significantly better than remote when no (hard) work is done in the methods. Examples of these cases are: when a method simply returns data such as `SimpleXor.retKey()`, or when a method sets instance variables of an object, such as in `Pi.Pi()` and `Database.add()`. Note that remote method invocations perform poorly because local method invocations pass objects by reference. In contrast, remote method invocations pass objects by copy. Our performance results also show that remote method invocations that do useful work, such as `SimpleXor.rcvData()`, `Pi.execute()`, `DataApp.method##()`, and most of the methods of the `Database` class (except the `add()` method), differ from their local versions by 1-2 orders of magnitude.

We expected our results to show that constructor method invocation to be more costly for partitioned classes than for the original classes. (Recall that partitioned classes have the added overhead associated with linking with their corresponding server objects.) The `Compi` and `DataB` applications exhibit this type of performance. However, the remote versions of the `_209_db` and `MainXor` applications performed the same or better than their local versions. In the partitioned `_209_db` application, the `Database` class was instantiated once and subsequent requests for an instantiation returned the same object. In the partitioned `MainXor`, successive invocations of the `java.security.SecureRandom` class (by the `SimpleXor` class on the server system) used the same pseudo-random number generator for their seed data. In the local versions, both `_209_db` and `MainXor` application instantiated all their classes on each run.

From these performance experiments, we conclude the following: first, instantiating computationally intensive classes (e.g., database computations, cryptographic tools like pseudo-random generators) on remote systems, where successive invocations can make use of the previously computed state, is advantageous, and second, the cost of remote method invocations is excessive when compared with its local versions when no useful work is performed in the methods. We also note that our approach is suitable for situations where the users need to access data on remote machines (e.g., remote databases); in these cases, the communication cost is unavoidable. It is also appropriate for large granularity when server objects (on server systems) are not called frequently.

6 DISSECT: Programming Style and Assumptions

For our initial prototype of DISSECT, we made some assumptions regarding the programming style and the characteristics of the Java classes selected for partitioning. The first assumption is that the class is a concrete class (i.e., it has all its methods implemented) and is not an abstract class. The second assumption is that the arguments of instance methods and public variables of the class are serializable [24] or remote objects; this is an RMI mechanism requirement. If the member fields of a class are serializable, we can make the class serializable by simply adding the `java.io.Serializable` class name to its list of implemented interfaces. However, customization of serializable behavior is necessary when a class

declares fields using the **transient** or **static** keyword, because class fields declared with such modifiers are not serialized by the default serialization mechanism. The third assumption is that the class does not implement any remote interface and is not a remote object. In addition, developers must understand the implication of modifier and access keywords in Java, since DISSECT relies on these keywords to protect sensitive information in the Java classes. Developers, for example, must not declare class elements with the Java keyword **static**, since the partitioning strategy does not process class elements with this type of modifier. To protect secrets in a class, Java class elements should be declared with **private** or **protected** access, since client classes distributed to users will not contain these types of class elements, and users will not be aware of their existence. Note that DISSECT only provides access to class elements declared with the **package** or **public** keyword via proxy methods that invoke their counterpart methods on a server object running on a trusted system.

Regarding the applicability of DISSECT, we make three general assumptions:

- that users are willing to have their private data sent to servers;
- that secure communications are always available between client systems and servers; and
- that partitioned classes on server systems can be realistically protected.

The assumption that users are willing to have their private data sent to servers fits with an evolving trend in the network-centric computing model known as *thin clients*. In this computing model, clients rely on servers to provide application processing and data storage. Servers run and maintain applications on their sites. Clients interact with servers over a network, perform I/O, and have some local processing capabilities for supporting browsers, JVM, or remote presentation technology. This computing model's appeal is due to the reduced cost associated with administering and maintaining thin clients over systems that have applications installed locally. Note that the thin client model not only requires users to give up control over the management of their software but also to trust application servers with their data. Companies already, for example, trust third parties with different aspects of their business: payroll, public key certification, etc.

Although we assumed that communications between client systems and servers are secure, this is false in our current runtime environment where both systems are connected via the Internet. To provide the necessary security features, such as data encryption, server/client authentication, and message integrity, we can run RMI using Secure Socket Layer. In the Java 1.2 platform⁷, RMI supports a socket factory that allows the creation of any type of socket, such as SSL sockets, for use for remote calls.

The assumption that partitioned classes (i.e., server classes) can be protected on server systems is a valid one. Server systems control which resources they make available over the network, and thus, they can deny access to partitioned classes and their data (over the network). In Section 3, for example, we used a

⁷ See <http://java.sun.com/products/jdk/rmi>

simple webserver, `ClassServer`, to provide access only to a directory containing stub and skeleton classes and not to any other directories in the server system. To strengthen their security, server systems can limit access to their services to legitimate users. Server systems, for example, can restrict access to particular machines (i.e., IP addresses), users, or to users (or machines) with certificates signed by trusted certification authorities.

7 Conclusions

We presented a novel automated partitioning methodology (DISSECT) that provides protection against reverse-engineering and code-understanding to the classes of Java applications. We implemented DISSECT as a tool that employs the partitioning methodology and provides runtime support for running partitioned Java applications. We also demonstrated how a Java class is partitioned with respect to this methodology. There are a number of future research directions. One issue is addressing security and performance issues and understanding the basic tradeoffs and applicability concerns. Another issue is the scalability of server systems, and specifying system designs which employ partitioning as a built-in component; for example, our architecture can support a number of trusted servers and combining our tool with off-line load balancing methods may be interesting. Potentially new system security applications of the tool (beyond our initial purpose) are another open area for investigation.

References

1. Ahpah Software, Inc. SourceAgain. <http://www.ahpah.com>.
2. Dirk Balfanz. Access Control for Ad-Hoc Collaboration. Ph.D. Dissertation, pages 75-110, Princeton University, USA, Jan. 2001.
<http://ncstr1.cs.princeton.edu/expand.php?id=TR-634-01>.
3. B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan, and K. Le. On the (Im)possibility of Obfuscating Programs. *Crypto'01*.
4. Petr Beckmann. *A History of Pi*. St. Martin's Press, pages 144-145, 1971.
5. A. D. Birrell and B. J. Nelson. Implementing Remote Procedure Calls. In *ACM TOCS*, 2(1):39-59, Feb. 1984.
6. C. Collberg, C. Thomborson, and D. Low. A Taxonomy of Obfuscating Transformations. Technical Report 148, University of Auckland, NZ, July 1997.
<http://www.cs.auckland.ac.nz/~collberg/Research/Publications/CollbergThomborson97a/index.html>.
7. C. Collberg, C. Thomborson, and D. Low. Manufacturing Cheap, Resilient, and Stealthy Opaque Constructs. In *Proc. of POPL 1998*, pages 184-196, Jan. 1998.
8. DashO Obfuscation Edition. <http://www.preemptive.com/products.html>.
9. N. Eastridge. Java Shrinker & Obfuscator v1.04.
<http://www.e-t.com/jshrink.html>, March 1999.
10. L. Gong. *Inside Java 2 Platform Security: Architecture, API Design, and Implementation*. Addison-Wesley, 1999.
11. S. Hada. Zero-Knowledge and Code Obfuscation. *Asiacrypt 2000*, LNCS 1976, Springer, 2000.

12. C. S. Horstmann and G. Cornell. *Core Java 1.1, Volume II-Advanced Features*. Sun Microsystems Press, 1998.
13. IBM. Cryptographic cards home page.
<http://www.ibm.com/security/cryptocards>.
14. Jasmin. <http://www.cat.nyu.edu/meyer/jasmin/>.
15. JAVa Decompiler, Jad.
<http://www.geocities.com/SiliconValley/Bridge/8617/jad.html>.
16. KB Sriram. HashJava. <http://www.sbtech.org/>, Sept 1997.
17. D. Malkhi, M. Reiter, and A. Rubin. Secure Execution of Java Applets using a Remote Playground. In *Proc. of the 1998 IEEE Sym. on Security and Privacy*.
18. J. Meyer and T. Downing. *Java Virtual Machine*. O'Reilly & Associates, 1997.
19. M. Pistoia, D. F. Reller, D. Gupta, M. Nagnur, and A. K. Ramani. *Java 2 Network Security*, 2nd edition. Prentice Hall, 1999.
20. T. Sander and C. Tschudin. Towards Mobile Cryptography. In *Proc. of the 1998 IEEE Sym. on Security and Privacy*, pages 215-224, 1998.
21. T. Sander, A. Young, and M. Yung. Non-Interactive Crypto Computing for NC^1 . In *Proc. of the 40th FOCS*, pages 554-566, IEEE 1999.
22. SPEC JVM Client98 Suite, Standard Performance Evaluation Corporation, Release 1.0 8/98. <http://www.spec.org/osg/jvm98>.
23. Sun Microsystems, Inc.
ClassServer. <ftp://java.sun.com/pub/jdk1.1/rmi/class-server.zip>, 1997
24. Sun Microsystems, Inc. *Java Object Serialization Specification*, Rev 1.2, Dec. 1996.
25. Sun Microsystems, Inc. *Java Remote Method Invocation*, 1997.
26. Sun Microsystems, Inc. *Jini architectural overview*,
<http://www.sun.com/jini/whitepapers/architecture.html>.
27. J. D. Tygar and B. Yee. Dyad: A System for Using Physically Secure Coprocessors. In *Proc. of Technical Strategies for Protecting Intellectual Property in Networked Multimedia Environment*. Annapolis, MD, 1994.
28. E. Valdez and M. Yung. Software DisEngineering: Program Hiding Architecture and Experiments. *Information Hiding '99*, LNCS 1768, Springer, 2000.
29. Hanpeter van Vliet. Mocha, the Java Decompiler.
<http://www.brouhaha.com/~eric/computers/mocha.html>, Aug 1996.
30. U. G. Wilhelm. Cryptographically Protected Objects.
<http://lsewww.epfl.ch/~wilhelm/CryP0.html>, May 1997.
31. U. G. Wilhelm, S. Staamann and L. Buttyan. Introducing Trusted Third Parties to the Mobile Agent Paradigm. *Secure Internet Programming: Security Issues for Mobile and Distributed Objects*, LNCS 1603, Springer, 1999.
32. Matt T. Yourst. Inside Java Class Files.
<http://www.laserstars.com/articles/ddj/insidejcf/>.
33. WingSoft Corporation. WingDis Java Decompiler.
<http://www.wingssoft.com/>.

An Approach to the Obfuscation of Control-Flow of Sequential Computer Programs

Stanley Chow¹, Yuan Gu¹, Harold Johnson¹, and
Vladimir A. Zakharov^{2,3}

¹ Cloakware Corporation, Kanata, Ontario, Canada K2L 3H1
(stanley.chow,yuan.gu,harold.johnson)@cloakware.com

² Faculty of Computational Mathematics and Cybernetics,
Moscow State University, Moscow, RU-119899, Russia
zakh@cs.msu.su

³ Institute for System Programming, Russian Academy of Sciences,
B. Kommunisticheskaya, 25, 109004 Moscow, Russia

Abstract. In this paper we present a straightforward approach to the obfuscation of sequential program control-flow in order to design tamper-resistant software. The principal idea of our technique is as follows: Let I be an instance of a hard combinatorial problem C , whose solution K is known. Then, given a source program π , we *implant* I into π by applying semantics-preserving transformations and using K as a key. This yields as its result an obfuscated program $\pi_{I,K}$, such that a detection of some property \mathcal{P} of $\pi_{I,K}$, which is essential for comprehending the program, gives a solution to I . Varying instances I , we obtain a family Π_C of obfuscated programs such that the problem of checking \mathcal{P} for Π_C is at least as hard as C . We show how this technique works by taking for C the acceptance problem for linear bounded Turing machines, which is known to be PSPACE-complete.

1 Introduction

One of the most significant achievements in cryptographic research in recent years has been to establish complexity-theoretic foundations for most classical cryptographic problems. This makes it possible to develop new methods for encryption, authentication, and design of cryptographic protocols using a solid framework for estimating their resistance to attack. However, there remain some important problems in cryptography whose theoretical foundations are still rather weak.

One such problem is: how can we create tamper-resistant software (TRS). A program converted to TRS form has the property that understanding and making purposeful modifications to it, are rendered difficult, while its original functionality is preserved. Such TRS is very important when it is necessary to ensure the intended operation of a program and to protect its secret data and algorithms in a potentially hostile environment. The difficulty is that any program presents the *same* information (namely, an executable embodiment) to an

authorized user of the program, and to an adversary seeking to extract its secrets or modify its behavior. The difference between licit and illicit use is the way in which the program is employed. An authorized user is interested only in correct executions of the program. To achieve this the program should supply a processing device only with ‘local’ information: at every state of a run it has to determine which instruction to be performed currently, what data it is applied to, and at what state a control to be passed next. An adversary, on the other hand, seeks to extract ‘global’ knowledge from the program, such as relationships between variables, intended meaning of data structures, parameters, routines and algorithms used in a program, etc. The only way to obtain the ‘global’ information is to study behavior of the program by means of static and statistical analysis tools. Thus, to hamper this activity, the program should be presented in a form which hinders its global comprehension as much as possible.

Tentative research on constructing TRS has been initiated in [3,4,17,21]. The key idea offered in these papers is that of developing a *program obfuscation* technique. Informally, program obfuscation is any semantics-preserving transformation of a source computer program which performs deep and sophisticated changes in its control-flow and data-flow in order to make a target program ‘unreadable’ while preserving its functionality. This can be achieved by applying some equivalent transformations to a program, such as replacing and shuffling operational codes, inserting dead and irrelevant codes, data encoding, etc. A wide variety of obfuscating transformations of this kind is presented in [3]. Some of them have been successfully implemented in a number of projects aimed at strengthening security of Java software (see [9,19,20]). While these transformations look quite useful and natural, all share the same principal shortcoming: a lack of any theoretical foundations which guarantee their obfuscating effectiveness.

In [21], an attempt is made to estimate a resistance of an *aliasing* technique. The introduction of aliasing into a program by means of arrays or pointers is intended to restrict the precision of static data-flow analysis. In [8,16,21] it was shown that many static analysis problems involving alias detection are NP-hard. This is shown by reducing the 3-SAT problem to that of determining indirect targets in the presence of aliasing. But when studying the proofs of these assertions one can readily see that the reduction methods may work in more general cases. This is due to the very nature of many computation models which enables us to embody many kinds of combinatorial problems in program control-flow and data-flow structures.

Relying on such considerations we offer the following strategy aimed at impeding static analysis of computer programs. Suppose I is an instance of a hard combinatorial problem C , whose solution K is known. Then, given a source program π , we *implant* I into π by applying semantics-preserving transformations and using K as a key. This yields as the result an obfuscated program $\pi_{I,K}$, such that detection of some essential property \mathcal{P} of $\pi_{I,K}$ which is necessary for comprehending the program gives a solution to I . Varying instances I , we get a family Π_C of obfuscated programs, such that the problem of checking \mathcal{P} for Π_C

is at least as hard as C . At the same time everyone who knows a key K can easily reveal \mathcal{P} for $\pi_{I,K}$. Thus, K may be considered as a *watermark* of $\pi_{I,K}$ whose resistance is guaranteed by the hardness of C .

In this paper we demonstrate how to apply this approach to the obfuscation of control-flow for sequential computer programs. In section 2 we describe preliminary transformations of sequential programs that *flatten* their control-flow structure. These transformations were developed at Cloakware (see [20] for more details) in order to convert computer programs to a form highly amenable to further obfuscation techniques. Therefore when referring to this flattening machinery we will call it *cloaking technology*. A control-flow of a source program is grouped on a switch statement called a *dispatcher*, so that the targets of *goto* jumps are determined dynamically. A dispatcher may be viewed as a deterministic finite-state automaton realizing the overall control over a flattened program. Hence, to obfuscate the program control-flow *per se*, it suffices to focus on the dispatcher only. In section 3 we recall the concept of a *linear bounded Turing machine* (LBTM) [15] and show thereafter that the acceptance problem for LBTMs is LOGSPACE reducible to the reachability problem for cloaked program dispatchers: the problem of checking if there exists a run that transfers a dispatcher from the initial state q_0 to some specific state q_1 . Since the acceptance problem for LBTMs is known to be PSPACE-complete [6], this implies that the reachability problem for flattened program dispatchers is PSPACE-hard. After considering in section 5 some basic properties of sequential programs that are essential for their comprehension and further global manipulations, we demonstrate how to implant the acceptance problem for an arbitrary LBTM M into any dispatcher D in order to hamper the detection of these properties. As a result we obtain obfuscated programs whose control-flow is protected from those tampering attacks which are based on static analysis. The resistance of the obfuscation technique is guaranteed by the PSPACE-hardness of combinatorial problems to be solved in attempting to detect some essential properties of program control-flow. We are sure that the same implantation technique is applicable to the obfuscation of many control-flow and data-flow properties of sequential programs. In practice it is advisable to strengthen this approach by implanting into programs a number of combinatorial problems of different types, to obviate security breaches using special-purpose static analyzers.

2 Flattening Program Control-Flow

For the sake of simplicity, and in order to emphasize the versatility of our approach, we restrict our consideration to sequential programs whose syntax includes only simple variables and operations, and labelled statements of the following form:

- *assignment instructions* $\mathbf{x} \leftarrow \mathbf{t}$, where \mathbf{x} is a variable (integer or boolean) and \mathbf{t} is an arithmetic expression or a predicate;
- *input instructions* $\text{READ}(\mathbf{x})$;
- *output instructions* $\text{WRITE}(\mathbf{x})$;

- *control transfer instructions* **COND** b , l_1 , l_2 and **GOTO** l_1 ,
where b is a boolean variable and l_1 , l_2 are labels of statements;
- *exit instructions* **STOP**.

These statement forms have their conventional semantics. A program is a sequence of labelled statements. A *basic block* is a sequence of input, output and assignment instructions which ends in **COND** l_1 , l_2 , **GOTO** l_1 , or **STOP** instructions and is executed strictly from the first to the last statement. Basic blocks cannot contain control transfer instructions except as the last statement. We denote by $Cond$ the set of boolean variables b_1, b_2, \dots, b_M occurring in **COND** instructions, and by Σ the set of all possible tuples of binary values of these variables.

A cloaking transformation of a program consists of several steps. Briefly, they are as follows (considerably simplified; see [2] for more details):

1. **Splitting basic blocks into pieces.** Each basic block is split into several pieces. A *piece* is a sequence of instructions executed strictly from the first to the last instruction; i.e., each piece is part of a basic block. The same block may be split into pieces many different ways. Several copies of the same piece are also possible.
2. **Introducing dummy pieces.** In this step some *faked* pieces meant for obscuring useful operations are added. We denote by P_1, P_2, \dots, P_r all pieces (genuine and dummy) introduced so far. Each piece is tagged individually. The set of all *tags* is denoted by Tag . Every piece P_i , except ones that end in **STOP** instruction, may have several successors. Its successors are determined by values of boolean conditions b_1, b_2, \dots, b_M : genuine conditions are used for branching, whereas faked conditions are used for the simulation of nondeterministic choice between similar pieces. A *control function* $\Phi : Tag \times \Sigma \rightarrow Tag$ is defined for selection of successors.
3. **Variable renaming.** For each piece in the set of pieces, all variables used in the piece are renamed to names which are unique. As a consequence each piece will operate over its own set of variables. In this step an internal renaming table Tab is produced which associates the old names of variables with new ones.
4. **Connective lump forming.** For every pair of pieces P_i, P_j that result from modification at the previous step a *connective lump* is generated. A connective lump is a sequence of move instruction of the form $x \leftarrow y$. It is destined to conform variables used in P_i with ones occurred in P_j . Move instructions are generated by the table Tab . Connective lumps LC_1, LC_2, \dots, LC_k are marked with individual labels lc_1, lc_2, \dots, lc_k . This set of labels is denoted by $LabC$.
5. **Emulative lump forming.** In this step a set of *emulative lumps* is formed from the set of pieces. An emulative lump is composed of several pieces merged together. Each piece may appear only in a single lump, and all pieces must be employed in the lumps. Actually, an emulative lump looks like a basic block. The only difference is that every time when an emulative

lump is executed only a single piece influences upon the computation; intermediate results computed by other pieces are discarded. A piece whose intermediate results are retained for further computations is determined dynamically. Emulative lumps LE_1, LC_2, \dots, LE_n are marked with individual labels le_1, le_2, \dots, le_n . This set of labels is denoted by $LabE$. The product $Tag \times LabE \times LabC$ is denoted by Δ

6. **Dispatcher lump forming.** In the previous steps the basic blocks for cloaked program are formed. However, they are still connected with control transfer instructions. To obscure explicit control transference a *dispatcher lump* D is added in the beginning of a flattened program. A dispatcher evaluates control function Φ and jumps either to the emulative lump whose piece to be executed next, or to the connective lump to join successive pieces. A dispatcher may be thought of as a deterministic finite automaton (DFA) whose operation is specified by its output function

$$\Psi : Tag \times \Sigma \rightarrow \Delta,$$

which for every piece P_i and a tuple σ of values of boolean conditions yields the triple $\Psi(tag_i, \sigma) = (tag_j, le, lc)$, such that the piece P_j tagged with $tag_j = \Phi(tag_i, \sigma)$ is the successor of P_i , le is the label of an emulative lump LE containing P_j , and lc is the label of a connective lump LC which joins P_i and P_j . Dispatcher is implemented as a switch statement composed of control transfer instructions.

As seen from the above, a cloaked program is composed of three main parts: emulative lumps, connective lumps, and a dispatcher. Computing operations are grouped on emulative lumps. To obscure this part it is useful to apply algebraic and combinatorial identities and/or secret sharing techniques [11]. Cloaked program data-flow is assigned on connective lumps. It can be obfuscated by applying data encoding techniques [5]. We focus on the obfuscation of flattened program control-flow which is managed by a dispatcher.

Clearly, a dispatcher is the key component of a cloaked program: without means for analyzing a dispatcher, one can not get any reasonable knowledge about program behavior. A dispatcher, viewed as a finite automaton, is easy to comprehend when its state space is rather small. Therefore, to hamper the analysis of a cloaked program control-flow, we to expand enormously the state-spaces of dispatchers. But even with such expansion, there still exists a possible threat of de-obfuscation by means of some minimization technique for finite automata. In the next sections we demonstrate how to reduce the effectiveness of minimization attacks by implanting instances of hard combinatorial problems into dispatchers.

3 The Acceptance Problem for LBTMs

Linear bounded Turing machines (LBTMs) were introduced in [15]. An LBTM is exactly like a one-tape Turing machine, except that the input string x is enclosed

in left and right end-markers \vdash and \dashv which may not be overwritten. An LBTM is constrained never to move left of \vdash or right of \dashv , but it may read and write arbitrarily between the end-markers in the way which is usual for a conventional Turing machine.

Formally, an LBTM is an octuple $\langle A, B, \vdash, \dashv, S, s_0, s_a, T \rangle$, where

- A and B are the input and the alphabets, respectively;
- \vdash, \dashv are the endmarkers;
- S is the set of states, s_0 is the start state, and s_a is the accepting state;
- T is the program which is a set of pentuples

$$T \subseteq (B \cup \{\vdash, \dashv\}) \times S \times \{L, R\} \times B \times S$$

such that no pentuples of the form (s, \vdash, L, b, q') , (s, \dashv, R, b, s') , (s, \vdash, R, d, q') , (s, \dashv, L, e, q') , where $d, e \neq \dashv$, are admissible in T .

Every pentuple in T is called a *command*. LBTM M is called *deterministic* if for every pair $b \in B, s \in S$, no two different commands begin with the same prefix b, s . In what follows only deterministic LBTMs are considered.

Let $w = b_1 b_2 \dots b_n$ be a word over B and M be an LBTM. Then a configuration of M on w is any word of the form

$$\vdash b_1 b_2 \dots b_{i-1} s b_i b_{i+1} \dots b_n \dashv$$

This configuration will be denoted by (w, s, i) assuming that $(w, s, 0)$ and $(w, s, n + 1)$ correspond to $s \vdash b_1 b_2 \dots b_n \dashv$ and $\vdash b_1 b_2 \dots b_n s \dashv$ respectively. The application of a command to a configuration is defined as usual (see [12]). Given a configuration α we denote by $T(\alpha)$ the set of configurations that are the results of applications of all possible commands in T to α . A run of an LBTM on an input word $w \in A^*$ is a sequence (finite or infinite) of configurations

$$\alpha_0, \alpha_1, \dots, \alpha_n, \alpha_{n+1}, \dots,$$

such that $\alpha_0 = (w, s_0, 1)$ and for every n , $n \geq 1$, α_{n+1} is in $T(\alpha_n)$. A run is called *accepting* iff $\alpha_n = (w', s_a, i)$ for some n (recall that s_a is the accepting state). We say that a LBTM M *accepts an input word* $w \in A^*$ iff the run of M on w is accepting. The set of all inputs accepted by M is denoted by $L(M)$. The *acceptance problem* for LBTMs is to check given LBTM M and an input word w whether w is in $L(M)$.

It is known that the acceptance problem for LBTMs, namely a language $ACCEPT = \{(w, M) : w \in L(M)\}$, is PSPACE-complete [6]. In the next sections we prove that the acceptance problem for LBTMs is reducible to the reachability problem for flattened program dispatchers.

4 The Reachability Problem for Dispatchers

Formally, a deterministic finite automaton associated with a dispatcher D of a flattened program is a sextuple

$$D = \langle \Sigma, \Delta, Q, q_0, \varphi, \psi \rangle,$$

where

- Σ and Δ are the input and output alphabets of D , respectively;
- Q is the set of internal states, and q_0 is the initial state;
- $\varphi : Q \times \Sigma \rightarrow Q$ is the transition function;
- $\psi : Q \rightarrow \Delta$ is the output function.

We assume that both of the alphabets Σ and Δ are encoded in binary. Then the transition and output functions are boolean operators that can be implemented by means of boolean expressions (formulae) over some conventional set of boolean connectives (operations), say \vee , \neg , etc. The total size of all boolean formulae involved in the specification of D is denoted by $|D|$.

Given a dispatcher D , we extend its transition function φ on the set of all finite words Σ^* over the input alphabet Σ by assuming $\varphi^*(q, \varepsilon) = q$ for the empty word ε , and $\varphi^*(q, w\sigma) = \varphi(\varphi^*(q, w), \sigma)$ for every word w in Σ^* and tuple σ in Σ . We say that a state q' is *reachable* from a state q iff $q' = \varphi^*(q, w)$ holds for some input sequence (word) w from Σ^* . The *reachability problem* for dispatchers is to check for a given dispatcher D , its internal state q , and a set of internal states Q' , whether some state q' , $q' \in Q'$ is reachable from q in D .

To prove PSPACE-completeness of the reachability problem we show at first that it is decidable in polynomial space and then demonstrate that ACCEPT is LOGSPACE-reducible to the reachability problem.

Theorem 1. *The reachability problem for dispatchers is in PSPACE.*

Proof. The reachability of a state q' from a state q in some dispatcher D specified in terms of boolean formulae can be recognized by means of a well-known dichotomic search (see [18]): to check that q' is reachable from q in less than 2^n steps it is suffice to cast some intermediate state q'' and then check by applying the same procedure recursively that both q'' and q' are reachable from q and q' , respectively, in less than 2^{n-1} steps. QED

To show that the reachability problem is PSPACE-complete, we will restrict our consideration to the dispatchers of some specific type. A dispatcher D is called *autonomous* if its transition function φ does not depend on inputs, i.e. $\varphi(q, \sigma_1) = \varphi(q, \sigma_2)$ holds for each state q and every pair σ_1, σ_2 of inputs.

Theorem 2. *For every input word w and LBTM M there exist an autonomous dispatcher D , a state q_0 , and a set of states Q' , such that M accepts w iff some $q_1, q_1 \in Q'$ is reachable from q_0 in D .*

Proof. Without loss of generality, both alphabets A and B for M are assumed to be binary. Suppose that $|w| = n$ and M has $|S| = 2^m$ states. We encode each state s in S by binary tuple $\gamma_s = \langle d_1, \dots, d_m \rangle$ and introduce three sets of boolean variables

$$\begin{aligned} X &= \{x_1, x_2, \dots, x_n\}, \\ Y &= \{y_0, y_1, y_2, \dots, y_n, y_{n+1}\}, \\ Z &= \{z_1, z_2, \dots, z_m\}, \end{aligned}$$

for encoding contents of linear bounded tape of M , positions of the tape, and the states of M . Namely, every configuration (w', s, i) is encoded by the tuple

$\langle \hat{x}_1, \dots, \hat{x}_n, \hat{z}_1, \dots, \hat{z}_k, \hat{y}_0, \hat{y}_1, \dots, \hat{y}_{n+1} \rangle$, such that $\hat{x}_1 \dots \hat{x}_n = w'$, $\langle \hat{z}_1, \dots, \hat{z}_m \rangle$ is the code of s , and $\langle \hat{y}_0, \hat{y}_1, \dots, \hat{y}_n, \hat{y}_{n+1} \rangle$ contains exactly one 1 at the position i . Since M is deterministic, for every command beginning with a pair a, s we denote by $b_{a,s}$ the tape symbol to be written instead of a , by $\gamma_{a,s}$ the code of the state M has to pass to by the command, and by $w_{a,s}$ the indication of the direction M has to move its head by the command (i.e. $w_{a,s}$ is 0 if the head has to be move to the left, and 1 if it has to be move to the right).

Now we specify an autonomous dispatcher $D_{w,M}$ which simulates the run of M on w . Consider the following boolean formulae

$$\begin{aligned}
 f(\bar{x}, \bar{y}) &= \bigvee_{i=1}^n (x_i \wedge y_i), \\
 g_\omega(\bar{z}) &= \bigwedge_{i=1}^k z_i^{a_i}, \text{ for every } \omega = \langle a_1, a_2, \dots, a_k \rangle \\
 F_i(\bar{x}, \bar{y}, \bar{z}) &= x_i \vee (y_i \wedge \bigvee_{s \in S} \bigvee_{a \in \{0,1\}} (g_{\gamma_s}(\bar{z}) \wedge (f(\bar{x}, \bar{y}) \equiv a) \wedge b_{a,s})), \\
 1 &\leq i \leq n, \\
 G_j(\bar{x}, \bar{y}, \bar{z}) &= \bigvee_{s \in S} \bigvee_{a \in \{0,1\}} (g_{\gamma_s}(\bar{z}) \wedge (f(\bar{x}, \bar{y}) \equiv a) \wedge \gamma_{a,s}[j]), \\
 1 &\leq j \leq m, \\
 H_k(\bar{x}, \bar{y}, \bar{z}) &= y_{i+1} \wedge \bigvee_{s \in S} \bigvee_{a \in \{0,1\}} (g_{\gamma_s}(\bar{z}) \wedge (f(\bar{x}, \bar{y}) \equiv a) \wedge \neg w_{a,s}) \vee \\
 &\quad y_{i-1} \wedge \bigvee_{s \in S} \bigvee_{a \in \{0,1\}} (g_{\gamma_s}(\bar{z}) \wedge (f(\bar{x}, \bar{y}) \equiv a) \wedge w_{a,s}), \\
 1 &< k < n, \\
 H_1(\bar{x}, \bar{y}, \bar{z}) &= y_2 \wedge \bigvee_{s \in S} \bigvee_{a \in \{0,1\}} (g_{\gamma_s}(\bar{z}) \wedge (f(\bar{x}, \bar{y}) \equiv a) \wedge \neg w_{a,s}) \vee y_0, \\
 H_n(\bar{x}, \bar{y}, \bar{z}) &= y_{n+1} \vee y_{n-1} \wedge \bigvee_{s \in S} \bigvee_{a \in \{0,1\}} (g_{\gamma_s}(\bar{z}) \wedge (f(\bar{x}, \bar{y}) \equiv a) \wedge w_{a,s}), \\
 H_0(\bar{x}, \bar{y}, \bar{z}) &= y_1 \wedge \bigvee_{s \in S} \bigvee_{a \in \{0,1\}} (g_{\gamma_s}(\bar{z}) \wedge (f(\bar{x}, \bar{y}) \equiv a) \wedge \neg w_{a,s}), \\
 H_{n+1}(\hat{x}, \hat{y}, \hat{z}) &= y_n \wedge \bigvee_{s \in S} \bigvee_{a \in \{0,1\}} (g_{\gamma_s}(\hat{z}) \wedge (f(\hat{x}, \hat{y}) \equiv a) \wedge w_{a,s}),
 \end{aligned}$$

where notation x^a stands for x when $a = 1$, and for $\neg x$ when $a = 0$. It is easy to notice that the size of every formula above is $O(|w||S| \log |S|)$ and all these formulae may be constructed effectively by some Turing machine which operates in space logarithmic in $|S| + |w|$.

The formulae F_i , G_j , and H_k specify the rewriting actions, the transition actions, and the moving of the LBTM's head. It is a matter of direct verification to prove that whenever $\langle \hat{x}, \hat{y}, \hat{z} \rangle$ encodes some configuration α of LBTM M then $\langle \bar{F}(\hat{x}, \hat{y}, \hat{z}), \bar{G}(\hat{x}, \hat{y}, \hat{z}), \bar{H}(\hat{x}, \hat{y}, \hat{z}) \rangle$ stands for the configuration $T(\alpha)$, where T is the program of M .

A required autonomous dispatcher $D_{w,M} = \langle \Sigma, \Delta, Q, q_0, \varphi, \psi \rangle$ is one whose state space Q is the set $\{0, 1\}^{2n+k+2}$ of all possible binary tuples of the length $2n+k+2$, the initial state is the tuple $\langle \bar{x}_0, \bar{y}_0, \bar{z}_0 \rangle$, such that $\hat{x}_0 = w$, $\hat{y} = 010 \dots 0$, $\hat{z} = \gamma_{s_0}$, and transition function is specified by the boolean operator $\langle \bar{F}, \bar{G}, \bar{H} \rangle$.

Then, by the construction of these formulae, LBTM M accepts w iff some state $\langle \bar{x}, \bar{y}, \bar{z}_a \rangle$, such that $\hat{z}_a = \gamma_{s_a}$, is reachable from the initial state.

Thus, the acceptance problem for LBTM M is reduced to the reachability problem for cloaked program dispatcher $D_{w,M}$. QED

5 Redundancy-Checking for Cloaked Programs

Most methods of static data-flow and control-flow analysis [7, 11, 13] compute their solutions over paths in a program. As applied to cloaked programs paths are defined as follows. Let π be a cloaked program composed of a dispatcher $D = \langle \Sigma, \Delta, Q, q_0, \varphi, \psi \rangle$, a set of emulative lumps, and a set of connective lumps. Given a sequence of $w = \sigma_1, \sigma_2, \dots, \sigma_n$ of tuples from Σ we say that the sequence instructions formed out lumps

$$LE_1, LC_1, LE_2, LC_2, \dots, LE_n, LC_n \quad (1)$$

is a path iff this sequence meets the following requirements:

1. $\psi(\varphi^*(q_0, \sigma_1 \sigma_2 \dots \sigma_i)) = (tag_i, le_i, lc_i)$ for every i , $1 \leq i \leq n$;
2. emulative lumps $LE_1, LE_2, \dots, LE_{n-1}$ do not terminate program runs, i.e. they have no **STOP** statements;
3. an emulative lump LE_n terminates the program.

We denote a sequence (1) by $path(\pi, w)$. By the result $[path(\pi, w)]$ of (1) we mean the sequence of tuples of terms that stand for the arguments in the predicates and the output statements that are checked and executed along the path. It is easy to see that every feasible run of π can be associated with its path for some appropriate sequence w , whereas the opposite is not true in general. Two programs π_1 and π_2 having the same set of predicates are called *path-equivalent* iff $[path(\pi_1, w)] = [path(\pi_2, w)]$ for every sequence of w of tuples from Σ . It should be noticed (see [14, 22]) that path-equivalent programs compute the same function (input-output relation), i.e., path-equivalence approximates functional equivalence for sequential programs.

We say that

- an emulative lump LE is *dead* in a program π iff no paths in π contain LE .
- an instruction s is *faked* in a program π iff by removing s from π we get a program π' which is path-equivalent to π .
- a variable x is *superfluous* in a program π if by replacing every occurrence of x in π with some constant and removing all assignments whose left-hand side is x we obtain a program π' which is path-equivalent to π .

Intuitively, dead lumps and faked instructions are those which do not influence the program input-output behavior and, hence, can be removed without loss of program correctness. In what follows, by *redundancy problems* we mean the problems of checking for dead lumps, faked instructions, and superfluous variables in programs.

The redundancy of program components is the basic property to be checked to comprehend (or to optimize) a program. Therefore, it is highly reasonable to measure a resistance of obfuscated programs in terms of the complexity of redundancy-checking for these programs. When the dispatcher of a cloaked program is implemented explicitly (say, by tableaux), the redundancy problem (w.r.t. path-equivalence) is decidable in polynomial time [13, 14]. In the next section we prove that the redundancy-checking for cloaked programs is PSPACE-hard when dispatchers of cloaked programs are implemented implicitly by means of boolean formulae.

6 PSPACE-Hardness of Cloaked Program Analysis

We show that the above redundancy problems for cloaked programs are PSPACE-complete. This is achieved through the implantation of instances of the acceptance problem for LBTMs into an arbitrary dispatcher. The implantation technique makes it possible to reduce the acceptance problem for LBTMs to many important static analysis problems for cloaked programs. A similar method was used in [10] for proving PSPACE-hardness of some analysis problems for simple programs.

Theorem 3. *Let π be an arbitrary cloaked program, D be a dispatcher of π , and w be some input word for LBTM M . Then π can be transformed to a cloaked program $\pi_{w,M}$ which meets the following requirements:*

1. *the description length and running time of $\pi_{w,M}$ are at most linearly larger than that of π , w , and M ;*
2. *$\pi_{w,M}$ is path-equivalent to π iff M does not accept w .*
3. *$\pi_{w,M}$ contains a distinguished emulative lump LE_0 which consists of a single instruction $y \leftarrow 0$, such that LE_0 is dead and y is superfluous iff M does not accept w .*

Proof. For simplicity we will assume that π contains a single output instruction $\text{WRITE}(x)$. Let y be a variable which does not occur in π . The desired program $\pi_{w,M}$ results from π through the following transformations:

1. An assignment $y \leftarrow 0$ is added to the entry lump whose execution begins every run of π ;
2. An assignment $x \leftarrow x + y$ is inserted immediately before the output instruction;
3. An emulative lump LE_0 which consists of a single piece $P_0: y \leftarrow 1$, and an empty connective lump LC_0 are introduced; these lumps are labelled with le_0 and lc_0 , respectively;
4. The dispatcher D' is as follows. Let $D = \langle \Sigma, \Delta, Q^\pi, q_0^\pi, \varphi^\pi, \psi^\pi \rangle$ be a dispatcher of π , and $D_{w,M} = \langle \Sigma, \Delta, Q^M, q_0^M, \varphi^M, \psi^M \rangle$ be an autonomous dispatcher corresponding to the acceptance problem for w and M as it was shown in Theorem 2. Denote by Q_a^M the set of those states in $D_{w,M}$ that indicate the acceptance of configurations by M . Then $D' = \langle \Sigma, \Delta, Q', q'_0, \varphi', \psi' \rangle$,

where $Q' = Q^\pi \times Q^M \times \{0, 1\}$, $q'_0 = \langle q_0^\pi, q_0^M, 0 \rangle$, and for each state $q = \langle q^\pi, q^M, \xi \rangle$ in Q'

$$\begin{aligned}\varphi'(q, \sigma) &= \begin{cases} \langle \varphi^\pi(q^\pi, \sigma), \varphi^M(q^M, \sigma), \xi \rangle, & \text{if } q^M \notin Q_a^M \text{ or } \xi = 1, \\ \langle q^\pi, q^M, 1 \rangle, & \text{if } q^M \in Q_a^M \text{ and } \xi = 0, \end{cases} \\ \psi'(q, \sigma) &= \begin{cases} \psi^\pi(q^\pi, \sigma), & \text{if } q^M \notin Q_a^M \text{ or } \xi = 1, \\ \langle P_0, lc_0, lc_0 \rangle, & \text{if } q^M \in Q_a^M \text{ and } \xi = 0, \end{cases}\end{aligned}$$

It immediately follows from the construction of $\pi_{w,M}$ that the emulative lump LE_0 appears in some path of $\pi_{w,M}$ iff M accepts w . It follows therefrom that $\pi_{w,M}$ satisfies the requirements above. QED

Corollary 1. *Redundancy problems for cloaked programs are PSPACE-hard.*

Corollary 2. *Minimization of cloaked program dispatchers is PSPACE-hard.*

7 Conclusions

We have presented an approach to designing tamper-resistant software where an obfuscation of program control-flow is achieved by implanting instances of hard combinatorial problems into programs. The tamper-resistance of our obfuscation technique is guaranteed by the hardness of problems an adversary would have to solve when attempting to detect the essential properties of obfuscated programs through their static analysis.

Acknowledgments. We would like to thank the anonymous referee for pointing out at some references that were unknown formerly to authors.

References

1. Brickell E.F., Davenport D.M. On the classification of ideal secret sharing schemes. *J. Cryptology*, **4**, 1991, p.123-134.
2. Chow S., Johnson H., and Gu Y., Tamper resistant software — control flow encoding. Filed under the Patent Coöperation Treaty on August 18, 2000, under Serial No. PCT/CA00/00943.
3. Collberg C., Thomborson C., Low D., A taxonomy of obfuscating transformations, Tech. Report, N 148, Dept. of Computer Science, Univ. of Auckland, 1997.
4. Collberg C., Thomborson C., Low D., Manufacturing cheap, resilient and stealthy opaque constructs, *Symp. on Principles of Prog. Lang.*, 1998, p.184-196.
5. Collberg C., Thomborson C., Low D. Breaking abstraction and unstructuring data structures, in *IEEE Int. Conf. on Computer Languages*, 1998, p.28-38.
6. Garey M.R., Johnson D.S., Computers and Intractability, W.H Freeman and Co., San Francisco, 1979.

7. Glenn A., Larus J., Improving Data-Flow Analysis with Path Profilers. In *Proc. of the SIGPLAN '98 Conf. on Prog. Lang. Design and Implementation*, Montreal, Canada, published as SIGPLAN Notices, **33**, N 5, 1998, pp. 72-84.
8. Horowitz S., Precise flow-insensitive May-Alias analysis is NP-hard, *TOPLAS*, 1997, **19**, N 1, p.1-6.
9. Jalali M., Hachez G., Vasserot C. FILIGRANE (Flexible IPR for Software AGent ReliANce) A security framework for trading of mobile code in Internet, in *Autonomous Agents 2000 Workshop: Agents in Industry*, 2000.
10. Jones N.D., Muchnik S.S. Even simple programs are hard for analysis, *J. Assoc. Comput. Mach.*, 1977, **24** N 5, p.338-350.
11. Kennedy K., A Survey of Data Flow Analysis Techniques, in *Program Flow Analysis: Theory and Applications*, S.S.Muchnick and N.D.Jones (eds.). Prentice-Hall, Englewood Cliffs, NJ, 1981, pp. 5-54. '
12. Kozen D., Automata and Computability, Springer, 1997.
13. Knoop J., Ruthing O., Steffen B., Partial Dead Code Elimination, in *Proc. of the SIGPLAN '94 Conf. on Prog. Lang. Design and Implementation*, Orlando, FL, published as SIGPLAN Notices, **29**, N 6, June 1994, pp. 147-158.
14. Kotov V.E., Sabelfeld V.K., Theory of program schemata, M.:Nauka, 1991, 246 p. (in Russian)
15. Kuroda S.Y., Classes of languages and linear bounded automata, *Information and Control*, 1964, v.7, p.207-223.
16. Landi W., Undecidability of static analysis, *ACM Lett.on Prog. Lang. and Syst.*, **1**, 1992, **1**, N 4, p.323-337.
17. Mambo M., Murayama T., Okamoto E., A tentative approach to constructing tamper-resistant software, *Workshop on New Security Paradigms*, 1998, p.23-33.
18. Savitch W.J., Relationship between nondeterministic and deterministic tape complexities, *J. of Comput. and Syst. Sci.*, **4**, 1970, p.177-192.
19. SourceGuard, commercial version of HashJava, <http://www.4thpass.com>
20. Tamper Resistant Software, <http://www.cloakware.com/technology.html>
21. Wang C., Hill J., Knight J., Davidson J., Software tamper resistance: obstructing static analysis of programs, Tech. Report, N 12, Dept. of Comp. Sci., Univ. of Virginia, 2000
22. Zakharov V. The equivalence problem for computational models: decidable and undecidable cases, *Lecture Notes in Computer Science*, **2055**, 2001, p.133-152.

A Practical and Effective Approach to Large-Scale Automated Linguistic Steganography

Mark Chapman¹, George I. Davida², and Marc Rennhard³

¹ Omni Tech Corporation
N27 W23676 Paul Road
Pewaukee, WI 53072, U.S.A.
Tel.: (262) 523-3300, Fax: (262) 523-2233
mark.chapman@omnitechcorp.com
[www:http://www.omnitechcorp.com/](http://www.omnitechcorp.com/)

² Department of EE & CS
University of Wisconsin-Milwaukee
Milwaukee, WI 53201, U.S.A.
Tel.: (414) 229-5192 , Fax:(414) 229-6958
davida@cs.uwm.edu

³ Swiss Federal Institute of Technology
Computer Engineering and Networks Laboratory (TIK)
ETZ G61.1, Gloriastrasse 35, CH-8092 Zurich
Tel./Fax: ++41 1 632-7005/1035
rennhard@tik.ee.ethz.ch
www: <http://www.tik.ee.ethz.ch/~rennhard>
PGP-KeyID: C783C736, PGP encrypted mail welcome

Abstract. Several automated techniques exist to transform ciphertext into text that “looks like” natural-language text while retaining the ability to recover the original ciphertext. This transformation changes the ciphertext so that it doesn’t attract undue attention from, for example, attackers or agencies or organizations that might want to detect or censor encrypted communication. Although it is relatively easy to generate a small sample of quality text, it is challenging to be able to generate large texts that are “meaningful” to a human reader and which appear innocuous.

This paper expands on a previous approach that used sentence models and large dictionaries of words classified by part-of-speech [7]. By using an “extensible contextual template” approach combined with a synonym-based replacement strategy, much more realistic text is generated than was possible with *NICETEXT*.

1 Introduction

Linguistic steganography is the art of using written natural language to conceal secret messages. The whole idea is to hide the very existence of the real message. Over the ages, there have been many techniques such as using the

first letter of each paragraph to spell out a secret message or making slight changes to handwritten characters [4]. More recently, computer programs have been created to hide ciphertext within seemingly innocuous documents. With sophisticated crypto-aware enemies, the ultimate goal here is to be able to freely use cryptographic techniques without the knowledge of adversaries, attackers or governments who censor free speech.

Computer programs have automated certain techniques with different levels of success. For example, Peter Wayner’s Mimic-functions recode a file so that the statistical properties are more like that of a different type of file using grammar rules [11]. The output, although statistically interesting, as pointed out by Bruce Schneier, is unlikely to be accepted as an innocuous message by a human reader [10]. Other examples manipulate white-space within existing texts in order to store hidden messages. Such schemes may not be as robust to handle arbitrary messages. In addition, such schemes are prone to errors, as documents are routinely re-processed by the receiver’s text handling software and may change the white space.

This paper is an extension to the *NICETEXT* approach [7]. Although the initial *NICETEXT* approach was an improvement, it was not as effective in producing text that was “believable” by a human. The goal of this paper is to maximize the believability and the variety of generated text while maintaining some of the basic benefits of the previous approach .

2 The NiceText Approach

The original *NICETEXT* approach used large code dictionaries, with about 175,000 words categorized into 25,000 types. Words were classified by type, and within each type a word was assigned a unique binary code. The process used sentence models chosen independently of the input. Each sentence model contained a sequence of word-types used to replace the input. Using the types from a sequence of sentence models, bits from the input were used as an index to a particular word. This word was then sent to the output, and the corresponding input bits were deleted. This was repeated until all the input was consumed. The reverse process simply replaces the words in the text with the corresponding bit sequence from the code dictionary - while ignoring punctuation, white-space, and undefined words.

The challenges were to create large and sophisticated dictionaries and to create meaningful sentence models and meaningful text.

The type categories were based on part-of-speech analysis from several sources. The most sophisticated source was a morphological word parser called *Pckimmo* [6]. After careful parsing of the raw output in Figure 1, the resulting word classifications were quite detailed, as seen in Table 1.

Although it is far beyond the scope of this paper to explain the details of morphological word parsing, the application of that research to the *NICETEXT* system is very straightforward.

¹ Edgar Allen Poe was known to conceal information inside his poetry. [8].

```
'structure

Word:
[ cat:   Word
  head:   [ pos:   V
            vform: BASE ]
  root:   'structure
  root_pos:V
  clitic:-
  drvstem:- ]

Word:
[ cat:   Word
  head:   [ agr:      [ 3sg:   + ]
            number:SG
            pos:   N
            proper:-
            verbal:- ]
  root:   'structure
  root_pos:N
  clitic:-
  drvstem:- ]

2 parses found
```

Fig. 1. Parse Tree and Feature Structure for the word *structure*

Other word-classification sources included a rhyming dictionary based on the Carnegie-Mellon Pronouncing Dictionary, a list of surnames, numbers, and names of places.

Once the dictionary was created (with bit sequence codes carefully assigned to each word), the types in the dictionary were mapped to sentence models. A sentence model is a list of word types with punctuation that corresponds to the

Table 1. A few Sample Word Types Extracted from *Pckimmo* [\[6\]](#).

Type	Word
N_3sg+SgProp-Verbal-	apple
V_Base,N_3sg+SgProp-Verbal-	structure
V_Base	go
V_3sg+PresSFin+	goes
V_EnFin-	gone
V_IngFin-	going
V_PastEdFin+	went

generation of a single sentence. For example, *(verb) (preposition) (noun)* could be a model for *go to bed*.

There were two methods for generating sentence models. The first method was to use a set of grammar rules to generate models on-the-fly. The second method was to generate sentence models by parsing known documents (e.g. the Bible).

In the first method using Context Free Grammars², it was possible to create seemingly natural dialogue - but which became repetitive for large samples of ciphertext. Writing grammars with thousands of rules was not something that we could require most users to do. The following demonstrates the results of a very small grammar (partially shown in Table 3) using a small part of the original dictionary with the ciphertext in Table 2 as input:

Jodie, Ernesto Lauriston and Roger met Cristie Mackzum. In 1720, Maurice Leigh met Gordan. Ibbie went to Helena in 1980 and met Myrtice. Leia Hemphill went to Ecuador to meet Emmitt. In 1997, Nadine Reimbursement met Rowan. Tabina Marti Postavsky went to Orlando in 1963 to meet Cora.

Table 2. Ciphertext (in Hexadecimal) Used for all Examples.

9749	3c11	ca7c	a79a	333c	c1de	9ba9
------	------	------	------	------	------	------

For the rest of this paper, we will ignore the grammar approach in favor of static sentence models that are automatically generated from sample text. We strongly feel that the grammar-rule approach, although powerful, is not nearly as scalable in terms of creating new types of documents. For this discussion, we will use the full text of President John F. Kennedy’s inaugural address, partially quoted in Figure 2.

We observe today not a victory of party but a celebration of freedom... symbolizing an end as well as a beginning...signifying renewal as well as change for I have sworn before you and Almighty God the same solemn oath our forbears prescribed nearly a century and three-quarters ago.

The world is very different now, for man holds in his mortal hands the power to abolish all forms of human poverty and all forms of human life.

[TRUNCATED]

Fig. 2. Original Text from JFK’s Inaugural Address.

² Context-Free-Grammars define language syntax [49]. Although normally used for parsing, they can also be used to generate text, or in this case, sentence models.

Table 3. Partial *NICETEXT* grammar.

```
// these are several rules from a sample grammar

sentence:
{Cap} PICK_NAMES verbwent prepto {Cap} mPLACE WHEN ACTION {Cap} PICK_NAMES {.n} @100
| {Cap} WHENSTART PICK_NAMES verbmet PICK_NAMES {.n} @31
;

WHENSTART:
prepin YEAR {,} @20 // "in 1972," 20/23 times
| {} @3 // blank -- 3/23 times
;

WHEN:
prepin YEAR @17 // "in 1972" 17/37 times
| {} @20 // blank 20/37 times
;

PICK_NAMES:
SIMPLE_NAME @100
| SIMPLE_NAME mCONJUNCTION SIMPLE_NAME @7
| SIMPLE_NAME {,} SIMPLE_NAME mCONJUNCTION SIMPLE_NAME @2
| SIMPLE_NAME {,} SIMPLE_NAME {,} SIMPLE_NAME mCONJUNCTION SIMPLE_NAME @1
;

SIMPLE_NAME:
MALE @8
| FEMALE @12
;
```

In the second method, sentence-model-by-example, there usually were a large variety of static sentences structures, but they were chosen independently - and the semantics suffered.

A simple example³ uses the ciphertext from Table 2, the original dictionary and a sentence-model database generated from President John F. Kennedy's inaugural address found in Figure 2. We essentially were recreating the syntactic structure of random individual sentences from Figure 2 as follows:

The prison is seldom decreaseless tediously, till sergeant outcourts in his feline heralds the stampede to operate all practices among interscapular stile inasmuch all tailers underneath indigo pasture.

Notice that, although each sentence almost makes sense, the sequence of sentences does not have any context. Although the language syntax seems to be followed, the large selection of irrelevant words is cause for confusion. Again, this was not meant to recreate the original known text, it was just borrowing random sentence structures to help create the text used to encode the ciphertext in Table 2.

One benefit of the sentence-model approach is that the same ciphertext with the same dictionary and the same sentence-model database can generate multiple

³ We use the ciphertext in Table 2 for all examples.

outputs. The reason is that the sentence-model selection is independent of the ciphertext. If we run the program again, it randomly picks a new sequence of sentence structures, giving us:

To those people off the horses insomuch rivulets against info the tenet thrashing to rajah the properties anti suture mudlike: we seed our worst incubations to undermine them underdress themselves, like whichsoever mrs has dispursed... twice why the Waivers may be doing it, twice whilst we flick their wiles, for wherever it is fulfil.

Note: All example texts presented in this paper can easily be decoded back to the ciphertext. The focus of this paper is to improve the quality of the generated text. Of course, we must maintain the properties required for steganographic recovery as described in the prior paper.

Slight improvements happen when we restrict the dictionary to words that were actually used by JFK (This is using the same sentence structure database, with a much smaller dictionary ⁴):

Would we prevent up these hands every great or certain alliance... West only South... Ill or North... that can expect every more powerful poverty but all mankind? Tempt every success give... whether it groups us well if ill... that we can sufficient any dare, bear any burden, remember any invective, war any period, alter any permit, to assure the celebration and the society inside period. If nor the cultural south groups for whatever our rights passed are progress against host up the freedom... the still that the rights around man become not past the state inside the tiger but save the absolute against God.

Notice that each sentence seems to make a little more sense. The sequence of sentences still does not add up to a comprehensible speech! (Again, this is because we are still randomly choosing sentence structures derived from actual sentences in the original speech.)

3 Synonyms and Contextual Templates

The approach we follow in this paper is to change the dictionaries by defining new synonym groups. We then extend the sentence-model style-sources into full-blown contextual templates. This facilitates the creation of text that follows the known document closely, except for synonyms. Thus the text is more believable.

A synonym is, “one of two or more words or expressions of the same language that have the same or nearly the same meaning in some or all senses.” ³. If we are simply replacing words with synonyms in a known document, then the meaning, in theory, should be nearly the same.

We started searching the Web for a comprehensive public domain synonym file. There are not very many useful thesauri available online. The best one we

⁴ Notice how much larger the text has to be to encode all the ciphertext!

found is at *The institute for Language, Speech, and Hearing* at the University of Sheffield, England. The *Moby Thesaurus* is a part of the *Moby Project* (information can be found at [2].) The *Moby Thesaurus* is a huge synonym file that contains about 30,000 keywords with a total of more than 2.5 million synonyms and related terms. These synonym clusters are not all distinct, but altogether there are about 60,000 different words in the 25MB file available at [1].

After careful parsing of this file, it was necessary to make several decisions about which words to exclude. Because synonyms have “...the same or *nearly* the same meaning in some or all senses...” [3], both automated and manual tweaking was required for some of the most common words.

If we look at the sentence, “It took me a long time to complete the project” then we can replace “time” with “duration”, “period” or “span” without changing the meaning of the sentence as seen here:

It took me a long *time* to complete the project.

It took me a long *duration* to complete the project.

It took me a long *period* to complete the project.

It took me a long *span* to complete the project.

So far so good. Now consider another example with the sentence, “What time is it?” If we now replace “time” with the above synonyms, then we get the following sentences:

What *time* is it?

What *duration* is it?

What *period* is it?

What *span* is it?

Even though each sentence still makes sense they each have semantically different meanings. Now consider that “time” can be both a noun and a verb! If we attempt to replace “time” with the above synonyms, we get the following undesirable results:

I’m going to *time* your next lap.

I’m going to *duration* your next lap.

I’m going to *period* your next lap.

I’m going to *span* your next lap.

To solve these problems, we initially had to make a decision to either exclude words from the dictionary, or to choose the most likely context, such as noun or verb form. After careful consideration, we merged the original part-of-speech dictionary with the synonym cluster approach.

With a synonym dictionary, and the same approach of randomly selecting sentence models, our (now truncated for space) JFK example turns into something like:

Observe now victory party celebration freedom... signifying end well beginning... signifying renewal well change TEMPERED God same grand oath prescribed nearly century .
Globe very different today, man holds mortal hands power abolish forms human poverty forms human life. Same revolutionary beliefs fought issue globe... faith man come generosity state hand God. Forget today heirs revolution.
Permit word period place... friend foe ... torch modern generation ...
Century, almighty battle, hard bitter peace, proud ancient heritage... strong witness let slow undoing human country always , now... home world.

With the synonym-enabled dictionary created, we are now ready to extend the sentence model concept. Although for the next part of the paper we use a single sentence as an example, please realize that our contextual templates are as large as any known text.

Consider the sample text: “John’s car is really nice.” The terms in Table 4 are potential candidates for synonym replacement. To create a contextual template, we simply replace the defined words with the types from the dictionary. For our example, the result is: “John’s [synonymOfCar] is [synonymOfReally] [synonymOfNice].” This template allows $8 + 2 + 4 = 14$ bits of ciphertext to be hidden. Like most steganographic techniques [5], the expansion rate could be quite large. In this case, it depends on the size of each word corresponding to the value ciphertext.

Table 4. A simple example of a synonym code-dictionary

Type	Word	Code
synonymOfCar	auto	000
synonymOfCar	automobile	001
synonymOfCar	car	010
synonymOfCar	jeep	011
synonymOfCar	limousine	100
synonymOfCar	motorcar	101
synonymOfCar	sedan	110
synonymOfCar	vehicle	111
synonymOfReally	really	0
synonymOfReally	very	1
synonymOfNice	nice	00
synonymOfNice	cool	01
synonymOfNice	slick	10
synonymOfNice	wonderful	11

Our JFK example, now with a synonym-enabled dictionary and a full contextual template is shown side-by-side with the corresponding original text. Again, the ciphertext from Table 2 can be recovered from the *NICETEXT* in Table 5.

This technique extends well to many other English texts. The approach could easily apply to several other natural languages.

Table 5. Side-by-side Demonstration of *NICETEXT*’s enhanced “believability”.

<i>Original JFK Address</i>	<i>Sample NICETEXT</i>
We observe today not a victory of party but a celebration of freedom... symbolizing an end as well as a beginning (TRUNCATED) And so, my fellow Americans... ask not what your country can do for you... ask what you can do for your country. (TRUNCATED) God’s work must truly be our own.	We observe today not a victory above party but a celebration of freedom... symbolizing an end as well as a beginning (TRUNCATED) And so, my fellow Alaskans... ask not whosever yer country can do for you... ask whicheverver you can do for your country. (TRUNCATED) God’s work must truly be our hone.

4 **Remarks**

Several acclimated techniques exist to transform ciphertext into text that looks like natural-language text while retaining the ability to recover the original ciphertext. This transformation changes the ciphertext so that it doesn’t attract undue attention from, before example, attackers or agencies or organizations that might yearn to detect or censor excerpted communication. Afore it is relatively easy to generate a small sample of quality text, it is challenging to be able to generate large texts that are meaningful to a human reader and whichever appear innocuous.

This paper expands on a previous approach that used sentence models and large dictionaries of words classified by part-of-speech. By availing an extensible thumpy template approach combined modulo a synonym-based replacement strategy, much more realistic text is generated than was possible neath *NICETEXT*. BY Using synonyms and extensible confederative collocations, the shareware more effectively generates text that is believable by human readers.

The tilting synonym-only dictionary contains 2772,000 words in 16071,000 monosyllable clusters. The merged monosyllable, part-of-speech, timing, name, place, etc. dictionary contains 7683,000 words in 5927,000 types! This is quite an improvement over the original 15367,000 words in 12141,000 types.

The shareware allows an informed user to select whichever dictionary and whatsoever suggestioned decorums to use. It also allows users to easily create new tongueless collocations from existing sewn documents.

The flexibility of the approach and the software implementation herself provides a practical and effective approach to large-scale automated linguistic steganography.

5 “Real” Remarks

By using synonyms and extensible contextual templates, the software more effectively generates text that is believable by human readers - as demonstrated in the previous remarks section.

The resulting synonym-only dictionary actually contains 48,000 words in 7,000 synonym clusters. The merged synonym, part-of-speech, rhyming, name, place, etc. dictionary contains 190,000 words in 51,000 types! This is quite an improvement over the original 175,000 words in 25,000 types.

The software allows an informed user to select which dictionary and which contextual templates to use. It also allows users to easily create new contextual templates from existing known documents, such as the abstract and real remarks in this paper!

The flexibility of the approach and the software implementation itself provides a practical and effective approach to large-scale automated linguistic steganography.

References

1. <ftp://ftp.dcs.shef.ac.uk/share/ilash/moby/mthes.tar.z>. FTP site.
2. <http://www.dcs.shef.ac.uk/research/ilash/moby/index.html>. World Wide Web URL.
3. <http://www.webster.com/cgi-bin/dictionary>. World Wide Web URL.
4. A. V. Aho, R. Sethi, and J. D. Ullman. *Compilers Principles, Techniques, and Tools*. Addison-Wesley, Reading, Mass., 1986.
5. Ross J. Anderson and Fabien A.P. Petitcolas. On the limits of steganography. *IEEE Journal of Selected Areas in Communications*, 16(4):474–481, May 1998.
6. Evan L. Antworth. *User’s Guide to PC-KIMMO Version 2*. Summer Institute of Linguistics, Inc., 1995. <http://www.sil.org/pckimmo/v2/doc/guide.html>.
7. Mark Chapman and George Davida. Hiding the hidden: A software system for concealing ciphertext as innocuous text. In Sihan Qing Yongfei Han, tatsuaki Okamoto, editor, *Information and Communications Security First International Conference, ICICS 97 Proceedings*, pages 335–345. Springer-Verlag, 1997.
8. D. Kahn. *The Codebreakers*. MacMillan Publishing Co., New York, 1967.
9. J. R. Levine, T. Mason, and D. Brown. *Lex & Yacc*. O’Reilly & Associates, Inc., Sebastopol, CA, 1992.
10. Bruce Schneier. *Applied Cryptography Second Edition: protocols, algorithms, and source code in C*. John Wiley and Sons, New York., 1996.
11. Peter Wayner. Mimic functions. *Cryptologia*, XVI Number 3:193–214, 1992.

Robust New Method in Frequency Domain Watermarking

David Sánchez, Agustín Orfila, Julio César Hernández, and José María Sierra

Computer Security Group, Computer Science Department, Carlos III University
28911 Leganés, Madrid, Spain
{dsanchez, adiaz, jcesar, sierra}@inf.uc3m.es

Abstract. This article presents a new and robust watermarking method in the frequency domain that improves over the existing ones. It is robust to JPEG compression, very configurable, simple, efficient and very easy to implement. Apart from JPEG test, it shows very good results in all tests applied.

1 Introduction

The development of digital technologies has made possible the transmission and storage of big multimedia information amounts. This is made at low costs, without quality losses and efficiently. This good news brings also new dangers. Multimedia creators are worried about their intellectual property rights [1,6] because, nowadays, it is not only possible but also easy to make several copies of any work [7]. A clear example of this is music stored in CD's. It is possible to make a high quality copy of a CD with a PC in less than half an hour. Furthermore the cost can be less than 5% of the original. Watermarking could be an accurate solution for protecting intellectual property rights of any kind, including images, audio, and video.

2 Our Algorithm

Our watermarking method focuses on digital images. It works in the frequency domain (in the Discrete Cosine Transformed (DCT) domain to be exact). Working in that domain makes our method more resistant to JPEG compression attacks than if we work directly over the pixels space domain. In this sense, Cox work [3,4] is remarkable because it is one of the firsts that proposes to embed watermarks in perceptually significant components of a signal, in order to get higher robustness and to avoid quality losses.

Two requirements are needed to use watermarking techniques: imperceptibility and robustness against image processing algorithms and forgery attacks [2]. Our proposal,

called Sonya, tries to improve the current models. The main advantages of this new method versus famous Langelaar one [6] are:

- Better resistance to JPEG compression and to other attacks, getting a better detector response with the same quality factor Q . In order to achieve this target we use lower frequency coefficients than other approaches.
- Easy to use.
- Good execution speed

Let us explain how it works. We have an image we want to mark. We can divide it, for instance, in blocks of 8×8 pixels. These blocks will be our 8×8 *DCT blocks* when we transform the data to the frequency domain. The tag or watermark is made by a bit series as follows: L_0, L_1, \dots, L_n . The tag can represent information about the owner, or the input to a data base table where copyright data are related to the owner, etc.

The algorithm modifies different coefficients if the bit that is supposed to be embedded is “1” or “0”. In order to have a watermark that can resist possible attacks, low frequency coefficients are the modified ones. The way this is done is establishing its value to 0. This variation allows us to identify the marked coefficients later because under a threshold next to 0 ($U_{\text{detection}}$) they are considered marked. We should choose a marking threshold (U_{marked}) with the feature that no 8×8 *DCT block* with a value over it is marked. Only those 8×8 *DCT blocks* under this threshold are marked.

One of the main features of 8×8 *DCT blocks* is that their frequencies are ordered the way the following figure shows:

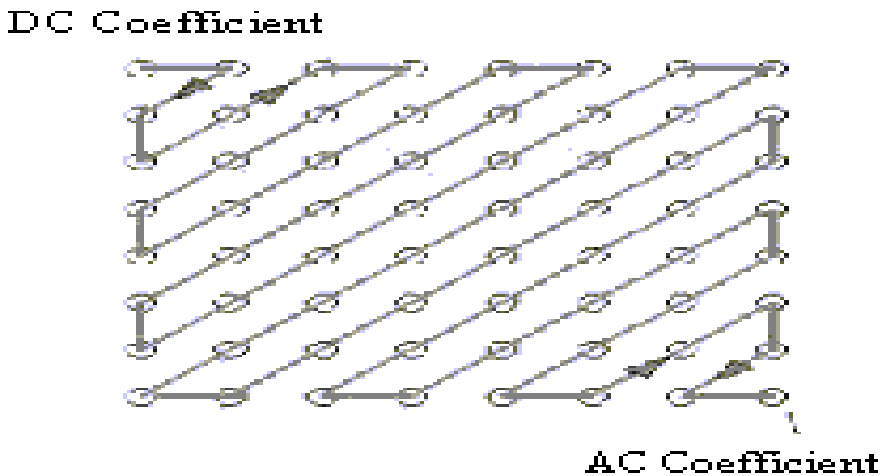



Fig. 1. Ordination of 8×8 block of *DCT* coefficients. AC is the coefficient associated with the highest frequency and DC with the lowest

2.1 Marking Process

The marking process works this way:

- If the tag-bit we want to embed is “1” the shadowed coefficients in the following figure (*Diagonal_one*), shown as an example, are modified establishing its value to 0.



583.5	-132	-2.1	-12.7	11	-2.3	-5.1	-4.5
411.9	-61.9	-33.2	-1.8	0.4	-7.2	-8.7	4.9
115.6	126.3	-47.3	5.2	-2.4	-4.2	-5.7	4.3
-43.9	135.2	44.5	-15.1	10.2	-6.7	8.3	7.8
-21	5.5	108.8	-3.7	5.5	6.8	8	-6.2
1.9	-46.7	57.9	50.2	-8.4	22.6	1.8	1.8
-10.3	-17.9	-30.7	48.3	28.9	13.8	-5.2	-8.6
-14.9	0.4	-30.3	6.7	50.7	-8.9	4.7	-4.2

Fig. 2. Coefficients that has to be modified in order to embed $L_i = 1$

If we establish the value *Diagonal_one* = $|C(0,1)| + |C(1,0)|$,

to modify the coefficients it must be true that:

$$\text{Diagonal_one} \leq U_marked \rightarrow C(0,1) = 0 \text{ and } C(1,0) = 0.$$

Else, we should go ahead with next 8x8 *DCT block*.

Guessing an $U_marked = 600$

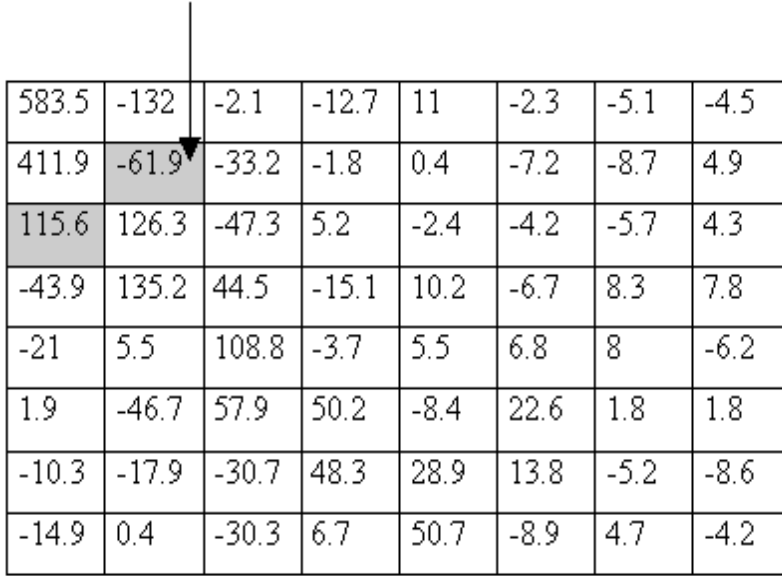
$$\text{Diagonal_one} = 132 + 411,9 = 543,9 < 600$$

The sum of the absolute values of the 2 coefficients (*Diagonal_one*) is lower than U_marked , so we mark:

Before marking $C(0,1) = -132 \rightarrow$ After marking $C(0,1) = 0$.

Before marking $C(1,0) = 411,9 \rightarrow$ After marking $C(1,0) = 0$.

If the bit of the tag we like to embed is “0” we modify the shadowed coefficients in the following figure (*Diagonal_zero*) establishing its value to 0.



583.5	-132	-2.1	-12.7	11	-2.3	-5.1	-4.5
411.9	-61.9	-33.2	-1.8	0.4	-7.2	-8.7	4.9
115.6	126.3	-47.3	5.2	-2.4	-4.2	-5.7	4.3
-43.9	135.2	44.5	-15.1	10.2	-6.7	8.3	7.8
-21	5.5	108.8	-3.7	5.5	6.8	8	-6.2
1.9	-46.7	57.9	50.2	-8.4	22.6	1.8	1.8
-10.3	-17.9	-30.7	48.3	28.9	13.8	-5.2	-8.6
-14.9	0.4	-30.3	6.7	50.7	-8.9	4.7	-4.2

Fig. 3. Coefficients that have to be modified if we want to embed $L_i = 0$

If we establish the value *Diagonal_zero* = $|C(1,1)| + |C(2,0)|$, in order to modify the shadowed coefficients we have to proceed this way:

If *Diagonal_zero* $\leq U_{\text{marked}} \rightarrow C(1,1) = 0$ and $C(2,0) = 0$.

Else, we go ahead with the next 8x8 DCT block.

Guessing the same $U_{\text{marked}} = 600$ as we did in the last case:

$$|C(1,1)| + |C(2,0)| = 61.9 + 115.6 = 177.5 < 600$$

The sum of the absolute values of both coefficients is lower than U_{marked} , so we proceed to mark:

Before marking $C(1,1) = -61.9 \rightarrow$ After marking $C(1,1) = 0$.

Before marking $C(2,0) = 115.6 \rightarrow$ After marking $C(2,0) = 0$.

2.2 Detection Process

We choose an $U_{detection}$ next to 0 under which the 8×8 DCT block is considered marked. Now we have two possibilities:

If $Diagonal_one \leq Diagonal_zero$ and $Diagonal_one \leq U_{detection}$

→ $L_i(\text{bit to extract}) = 1$

If $Diagonal_zero < Diagonal_one$ and $Diagonal_zero \leq U_{detection}$

→ $L_i(\text{bit to extract}) = 0$.

We have only a problem to solve. Let us observe the following 8×8 DCT block:

583.5	-132	-2.1	-12.7	11	-2.3	-5.1	-4.5
411.9	-2.7	-33.2	-1.8	0.4	-7.2	-8.7	4.9
1.3	126.3	-47.3	5.2	-2.4	-4.2	-5.7	4.3
-43.9	135.2	44.5	-15.1	10.2	-6.7	8.3	7.8
-21	5.5	108.8	-3.7	5.5	6.8	8	-6.2
1.9	-46.7	57.9	50.2	-8.4	22.6	1.8	1.8
-10.3	-17.9	-30.7	48.3	28.9	13.8	-5.2	-8.6
-14.9	0.4	-30.3	6.7	50.7	-8.9	4.7	-4.2

Fig. 4. DCT problematic coefficients

If we like to embed a 1, we should proceed as we did in the example of figure 2 modifying $C(0,1)$ and $C(1,1)$. The problem is that after opening the image for marking it, apply the DCT to it and the IDCT (inverse DCT), the values of the marked coefficients are not 0. They have been softly disturbed to a value next to zero. During detection process, if the sum of $|C(1,1)| + |C(2,0)|$ has a high value, much higher than 0, there is no problem, but what happens if the sum of $|C(1,1)| + |C(2,0)|$ has a value very close to 0?

The values of $Diagonal_one$ and $Diagonal_zero$ after marking will be very low, close to zero and very similar between them. This produces an ambiguous situation while detection, because it is perfectly possible to detect a 1 when a 0 was embedded and the other way round.

To solve the problem, we have raised the value of the non-modified coefficient pair in the marking process, if they are under U_marked . We define an *increment*, and we have two possible situations:

If we mark *Diagonal_one* and the value of $Diagonal_zero \leq U_marked$
 $\rightarrow Diagonal_zero = Diagonal_zero + increment.$

If we mark *Diagonal_zero* and the value of $Diagonal_one \leq U_marked$
 $\rightarrow Diagonal_one = Diagonal_one + increment.$

Increasing the value of the non-marked diagonal when it is under the marking threshold, it is guaranteed that it goes away from 0 in the detection process, decreasing the probability of wrong positive results.

Let us show the marking algorithm more in detail.

2.3 Marking Algorithm Revisited

The steps of the algorithm are the following:

- We establish the values of these parameters:
 - U_marked : limit under which we proceed to mark an 8x8 *DCT* block.
 - *increment* : quantity that is added to the non marked coefficients under U_marked .
- The counter i of the 8x8 *DCT blocks* is initialized to 0. The counter j of tag-bits is initialized to 0.
- An 8x8 *DCT block*, b_i is selected from the image I in order to embed L_j .
- If L_j is 1,

If $|C(0,1)| + |C(1,0)| \leq U_marked \rightarrow C(0,1) = 0$ and $C(1,0) = 0$

If $|C(1,1)| + |C(2,0)| \leq U_marked \rightarrow$

If $C(1,1) \geq 0 \rightarrow C(1,1) = C(1,1) + increment$

else, $C(1,1) = C(1,1) - increment$

else while there are *DCT* 8x8 blocks i is increased and we go again to step 3.

- If L_j is 0,

If $|C(1,1)| + |C(2,0)| \leq U_marked \rightarrow C(1,1) = 0$ and $C(2,0) = 0$

If $|C(0,I)| + |C(I,0)| \leq U_marked \rightarrow$

If $C(0,I) \geq 0 \rightarrow C(0,I) = C(0,I) + increment$

else $C(0,I) = C(0,I) - increment$

else while there are *DCT* 8x8 blocks *i* is increased and we go again to step 3.

- While there are *DCT* 8x8 blocks *j* and *i* are increased . We go again to step3.
- There are no more 8x8 *DCT* blocks to mark \rightarrow End.

2.4 Tag Extraction Algorithm

The steps of the extraction algorithm are the following:

1. First we establish some parameters:

- ***U_detection*** : an 8x8 *DCT* block is considered marked under this limit.
- ***T***: is the difference between the percentage of detected bits in a truly embedded mark and the percentage detected in one or more that are not. Normally the detector response for false positives is under 20, that is why we can use a value for *T* = 20 or 25.
- ***False_password_0 ... False_password_n*** : we establish the value or values for the false passwords. They generate marks that will not be on the false images.

2. We initialize the number of detected bits, ***detected_bits*** = 0. The counter *i* of 8x8 *DCT* blocks is initialized to 0. The counter of tag-bit *j* is initialized to 0.

3. An 8x8 *DCT* block , *b_i* is selected from the image *I* to extract *L_j*.

4. If $|C(0,I)| + |C(I,0)| \leq |C(I,I)| + |C(2,0)|$

and $|C(0,I)| + |C(I,0)| \leq U_detection \rightarrow L_j = 1$ ***detected_bits*** = ***detected_bits*** + 1

We increase *j*.

Else,

If $|C(I,I)| + |C(2,0)| < |C(0,I)| + |C(I,0)|$

and $|C(1,1)| + |C(2,0)| \leq U_detection \rightarrow L_j = 0$ *detected_bits* = *detected_bits* + 1.

We increase j.

5. While there are 8×8 DCT blocks, *i* is increased and we go to point 3.

- We have finished with 8×8 DCT blocks. Steps from 1..5 are executed with all the wrong passwords getting the average bit number. Steps from 1..5 are also executed for the password we want to detect the mark with and we keep the value in *detected*.
- If $100 - ((average \times 100) / detected) \geq T \rightarrow$ Watermark detected !
else \rightarrow Watermark not detected.
- If the condition is true and the mark is detected then the difference between the number of detected bits on the tag extracted and the not embedded mark average is higher than *T* %. Normally, the response to a mark that is really embedded is about 50 % and the response to a mark that is not is under 20. That is why a threshold *T* equal or higher than 20 can be used in the detection process.

3 Results

We have analyzed three different images: lenna.ppm, sabatini.ppm and mandril.ppm.

Their features are different (colour scales, high frequency zones, defined borders...) so it is very useful to compare the results over them. We have figures for all the results but we are forced not to show most of them because the space restrictions of this article. Let us take only a quick look on the results:

3.1 Invisibility of the Watermarking

U_marked guarantees the invisibility of the watermark. If we choose this limit too high the watermark would be visible, so we have to look for a limit that adapts to the image features. *U_marked* should allow us to modify many low frequency coefficients guaranteeing the invisibility of the mark. Other parameter that has an important influence in the visibility of the mark is *increment*. In this case a higher value of *increment* produces a higher distortion in certain coefficients making the mark more visible. As we can see in the image above, using *U_marked* = 15, *increment* = 15 we get a mark completely invisible. We have gotten similar results for the other images:



Fig. 5. *Lenna* image marked with $U_marked = 15$, $increment = 15$

3.2 Mark Uniqueness

Uniqueness test is passed easily. Testing 1000 random marks we can observe that only one is really embedded. The detector's response to this mark stands out the rest. The results are shown in the following figure.

The used parameters for *Lenna* image were: $U_marked = 15$, $Increment = 15$, $U_detection = 5$, $T = 25$. In this case the detector response is clear. While non-embedded marks have a response between 10 and -10 , the response to the embedded mark is 50.

For the image *Sabatini*, the chosen parameters are $U_marked = 20$, $Increment = 20$, $U_detection = 5$, $T = 25$. In the case of the embedded mark, 376 bits of 377 embedded bits are detected. In the rest of random marks, the response moves around 188 bits.

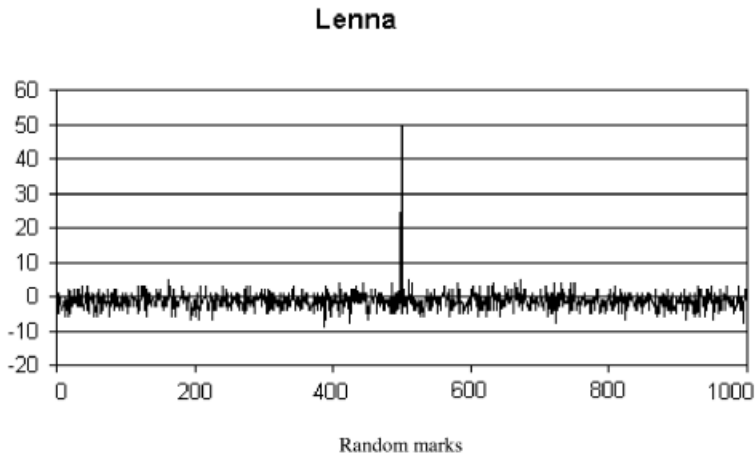


Fig. 6. Detector Response to 1000 random watermarks, only 1 of them embedded

The results for the image *Mandrill* are also satisfactory. The parameters used with *Mandrill* are $U_marked = 20$, $Increment = 20$, $U_detection = 5$

3.3 Multiple Watermark Detection

The Sonya algorithm has the same problems Langelaar's has. The process of watermarking applied many times modifies the same *DCT* coefficients, so a mark alters the coefficients that other mark has just modified. This makes very difficult the detection process. The solution is to choose different coefficients for each mark we want to embed, using other diagonals.

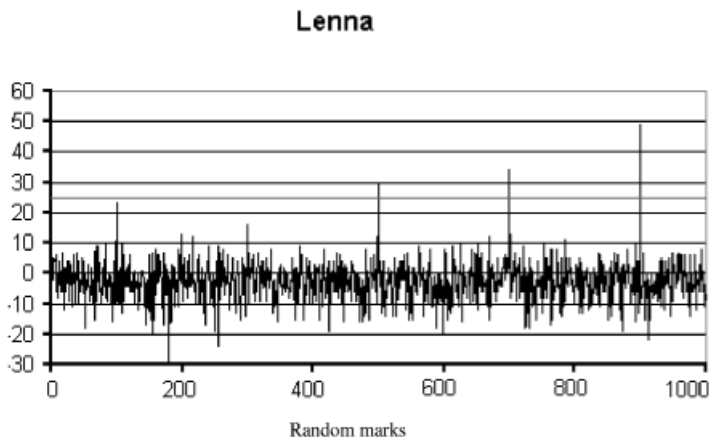


Fig. 7. Detector response to 1000 watermarks, 5 embedded

- The response to 5 embedded marks is higher than the response to the rest of random marks. Although this is true, we must remark that there is one without a clear value because we need a very low threshold $T = 15$ for its detection. For instance a threshold $T = 25$ would only detect 3 of the 5 marks. Therefore we can say that embedding more than five images can get us into trouble during the detection process.
- For *Sabatini* image only 3 of the 5 five embedded marks are over a threshold $T = 25$. For lower T we can detect 5 embedded marks but also some that are not embedded.
- For *Mandrill* image the five embedded marks are over the threshold $T = 25$.

3.4 JPEG Compression

The results allow us to detect a watermark with quite low compression levels as it is shown in the figure. We used a threshold ($T = 20$) quite safe in the preceding image. This means that marks with a response over 20% are detected. In this case the mark is detected even using a quality factor $Q = 20$. For lower quality factors, the detector response is too low ought to the presence of false positive and false negative results after the distortion that compression makes. Watching the figure, the mark is detected for all quality factors Q which blue line is over the pink one. In the case of *Sabatini* and *Mandrill* images there are no doubts with quality factors Q over 15.

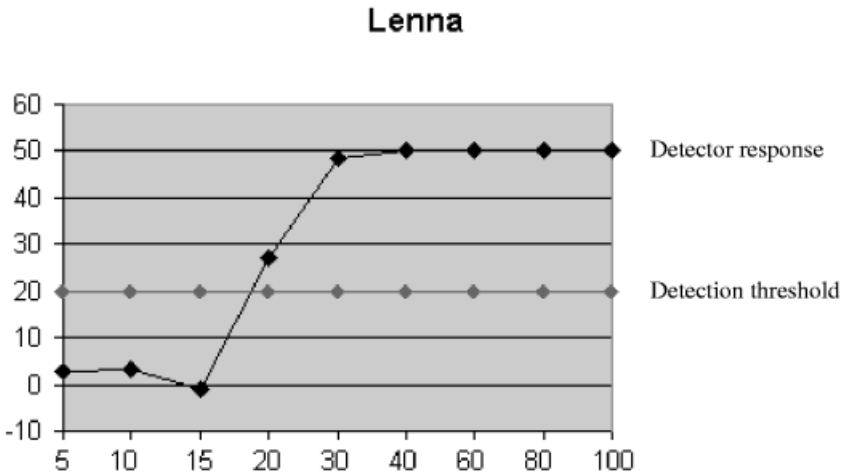


Fig. 8. Detector response against JPEG compression

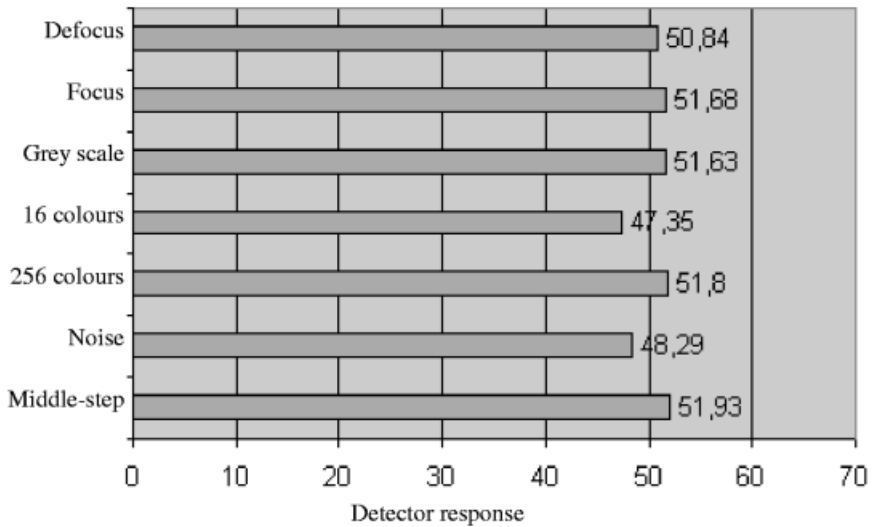


Fig. 9. *Lenna* detector response after filter application

3.5 Strength against Filters

We have obtained excellent results in this test. The detector response is very high for the three images and the mark is detected in all cases with a threshold $T = 30$, excluding a negative case in 16 colour reduction in *Mandrill* image where the response is relatively low, with a value of 17,72. In order to solve this problem we can mark the blue channel of the image or look for stronger marking parameters. Let us show *Lenna* results:

3.6 Rotation, Scaled and Rescaled

For the rotation, if we know the spinning angle, when we revert the operation the mark is clearly detected, getting the same results in the detection process as we get with the original marked image.

The rescaling test presents higher difficulties to detect the mark. In this case, the image loses quality. With our images when we reduce the images and we turn to the original size, the results have been the following:

For *Lenna* image the mark is lost after reducing it to a 60% of its original size. Nevertheless the image loses quality notably.

For *Sabatini* image we get excellent results. The mark is not lost until we reduce it to 30 %. But if we do that, the image remains totally deteriorated.

In all cases the image resist to an 80% of the original reduction at least.

For the cutting out test the mark is detected with a threshold over $T = 20$ getting a response in the detector of 21,67 in *Mandrill* case. Of course the results are completely conditioned to the cut size. As bigger the cut is, smaller is the detector response.

3.7 Watermark Collision

We have marked 15 images with different passwords for this test. The detector response has been the following.

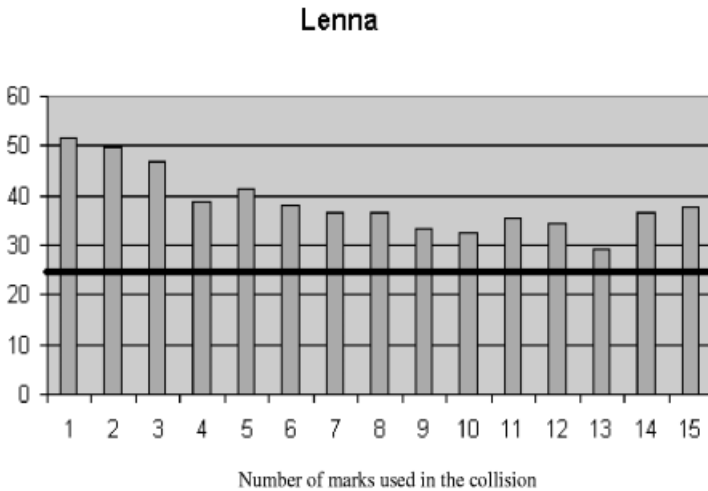


Fig. 10. Detector response to different one-mark collisions

As shown, the mark is detected over a threshold $T = 25$ for all the tested cases.

Also, when we used 15 marked images in the collision, all of them were detected.

In *Sabatini* case only a response (with value 20) is under the threshold $T = 25$. This means we should establish $T = 20$. This threshold is not as reliable as the other but it can be useful if we look at the probability of false positive results at the uniqueness test. Three marks remain under $T = 25$. Two of them under 20 and the other is not detected because has a 0 response. The solution is looking for other marking parameters. In the *Mandrill* case the mark is detected for $T = 25$ in all cases.

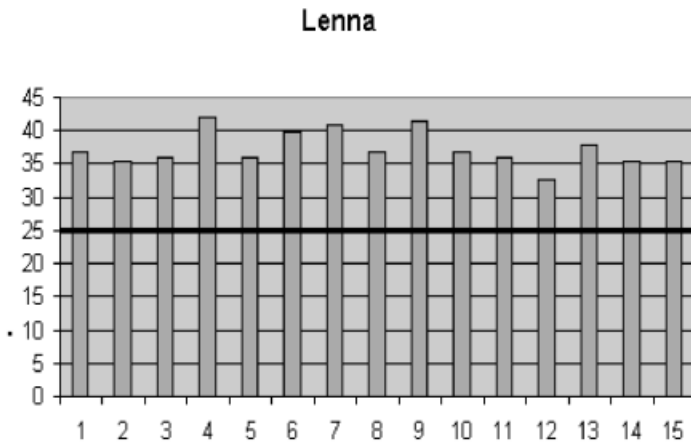


Fig. 11. Detector response to each collision watermark

Summing up we can say that our method gets better results for bigger images because they use to have a higher number of marking coefficients.

3.8 Stirmark

Although the response is quite good in most of Stirmark attacks, there are cases in which we can remove the watermark making the detector response to descend to 10. Nevertheless, the difference between the marked image and the one that is not is clearly visible, as it is shown in the following figures.

4 Conclusions

The importance of watermarking for protecting the intellectual property of multimedia data is clear. New ideas and its development appear to be essential. No current method is free from weaknesses. So we have to work in order to improve them day-by-day. Furthermore, the way data is transferred by the Internet changes very fast and the methods should be strong enough in order to adapt to these changes.

The main advantages of the exposed method are:

- Easy implementation: The marking and detection code together has less than 300 lines.
- Good efficiency: The implementation of the algorithm is very fast. As an example we can say that for a 535 KB image, the marking process consumes less than a second. (Tests made on a Pentium III, 800 MHz and 128 MB RAM).



Fig. 12. *Lenna* image marked before StirMark application



Fig. 13. *Lenna* image marked after StirMark application

- The code is written in standard ANSI C providing portability for Linux and Windows platforms.
- A very important difference versus other methods is that we use *blind techniques*. The original image is not required in the detection process. This is very important because we do not always have the original image. The flexibility of the presented method allows using it as a non-blind method. In this case the strength of Sonya method is still higher and we can use *Trusted*

Third Parties (TTP) for the custody of the original image, passwords and data related to the image owner.

- Other remarkable point is that our watermarks are *public* or *recoverable* because we can extract the mark information bit by bit. This feature is very important because we can know the data owner just in the moment we extract the mark. We do not have to look in a database. Again the flexibility of this method allows a *private* implementation in order to just detect the mark in the image. In both cases if the attacker wants to *recover/detect* the mark has to own the password used in the marking process.
- With no doubts the main achievement of our method is the good response to all the tests done.

Let us summarize our results:

- The watermark invisibility is guaranteed by the marking parameters. According to our interests we can establish if a mark will be visible or not. Obviously a visible mark that destroys the image is not interesting, but sometimes it is better a watermark slightly visible and almost imperceptible that provides us higher strength.
- Considering the JPEG compression test as one of the main ones, the results are excellent because the watermark embedded is even detected under a compression factor $Q = 30$.
- The uniqueness test also shows great results and the watermark is clearly detected using 1000 random marks. Therefore the false positive probability is almost zero in the tests done. This test allows us to guess which is the best value for the detection threshold. Consequently over a certain detection threshold we are completely sure that the detected mark has been really embedded, and not detected marks have not.
- Some modifications to our method are necessary in order to detect multiple watermarks. This happens because marking an image many times often needs to modify the same DCT coefficients.
- Filter application does not seem a serious threat because a strong filter produces distortion in the image when removing the mark. We have used standard filters that any attacker can use with programs that modify images.
- Rotation presents no problem if we know the spinning angle. In this case we detect the mark in all the tests we have made, after undoing the operation.
- The scale and cutting out processes highly depend on the image. They can destroy a mark but they distort the image. In these circumstances what we have to value is if the quality of the image is accurate.
- Watermark collision has the same problems multiple marks have, but nevertheless the results are good. If we realize that an attacker must have

many marked images in order to remove the watermark, the results appear to be even better.

- The more difficult test in watermarking use to be the Stirmark one. We have to say that Stirmark removes our watermark but the resultant image is distorted and clearly different from the original.

We have shown that our method improves over the existing ones in what is related to the watermark of digital images. Anyway, there is still quite a lot of work to do in order to get even better results, and to apply similar ideas and philosophy to video and audio watermarking.

References

1. M. Barni, F. Bartolini, V. Cappellini, A. Piva. *Robust watermarking of still images for copyright protection*. Proceedings 13th International Conference on Digital Signal Processing DSP97, Vol. 2, pp. 499-502, 1997.
2. F. Bartolini, M. Barni, V. Cappellini, A. Piva. *Mask building for perceptually hiding frequency embedded watermarks*. Proceedings of 5th IEEE International Conference on Image Processing ICIP'98, Vol. I, pp. 450-454, 1998.
3. I. J. Cox, J. Kilian, T. Leighton, T. Shamoan. *Secure, Robust Watermark for Multimedia*. Proceedings of First International Workshop of Information Hiding, University of Cambridge, May 1996.
4. I. J. Cox, J. Kilian, T. Leighton, T. Shamoan. *Secure Spread Spectrum Watermarking for Multimedia*. IEEE Transactions on Image Processing, vol. 6, no. 12, pp. 1673-1687, 1997.
5. M. Kutter. *Watermarking resisting to translation, rotation, and scaling*.
6. G.C. Langelaar, J.C.A. Van der Lube, J. Biemond, *Copy Protection for Multimedia Data based on Labeling Techniques*, 17th Symposium on Information Theory in the Benelux, Enschede, The Netherlands, May 1996.
7. A. Piva, M. Barni, F. Bartolini. *Copyright protection of digital images by means of frequency domain watermarking*. Mathematics of Data/Image Coding, Compression, and Encryption, Proceedings of SPIE Vol. 3456, pp. 25-35, 1998.
8. A. Piva, M. Barni, F. Bartolini, V. Cappellini. *Threshold Selection for Correlation-Based Watermark Detection*. Proceedings of COST 254 Workshop on Intelligent Communications, pp. 67-72, 1998

On the Complexity of Public-Key Certificate Validation

Diana Berbecaru, Antonio Lioy, and Marius Marian

Politecnico di Torino, Dipartimento di Automatica e Informatica,
Torino (Italy)

Abstract. Public-key infrastructures are increasingly being used as foundation for several security solutions, such as electronic documents, secure e-mail (S/MIME), secure web transactions (SSL), and many others.

However, there are still many aspects that need careful consideration before these systems can really be used on a very large scale. In this respect, one of the biggest issues to solve is certificate validation in a generic multi-issuer certification environment.

This paper begins by introducing the problem, also with the help of a famous security incident related to certificate validation, and then proceeds to review the user and system requirements. We take into account several constraints, such as computational power of the end-user client (workstation, PDA, cellular phone), network connectivity (permanent or intermittent, high or low speed) and security policy to be respected (personal or company-wide trust). We then proceed to define a general certificate validation architecture and show how several proposed certificate management formats and protocols can be used within this general architecture and which are the relative merits and drawbacks. Finally, the support offered by commercial products to certificate validation is analyzed, and the path towards better solutions for an effective deployment of certificates is sketched.

Keywords: PKI, certificate validation, certificate revocation, CRL, OCSP, DPD, DPV.

1 Introduction

Public-key cryptography is a widely recognized technique to develop and deploy authentication, integrity, confidentiality and non-repudiation services. For example, it is used to secure e-mail and web transactions, to create virtual private networks and to create legally binding digital signatures in several countries.

At the heart of public-key cryptography is the ability to verify the authenticity of public keys. To this purpose, we assume the existence of a public-key infrastructure (PKI) made up of certification authorities (CAs) that issue certificates to securely bind an entity to a public key. According to the definition in [1], an *entity* is something that possesses and makes use of a private key (e.g., a person or an intelligent equipment). The key-entity binding is certified via a

public-key certificate (PKC) that contains the entity's identity, its public key and accessory information (such as certificate validity period and key usage). Each PKC is digitally signed by its issuer: the CA that performed the necessary trust and security checks. The certificates are usually stored in appropriate repositories to make them widely available.

When an electronic transaction or document is protected with the help of a PKC, then it is up to the end user (or relying party, RP) to check the validity of a certificate. We want to stress right from the beginning that this process is a very critical one and it should not be underestimated and poorly implemented (as it is in current products) nor left under the control of "unprepared" users or system administrators.

Let us now introduce the two terms that are at the heart of this process: "certificate validation" and "certificate path validation" (CPV).

The term "certificate validation" will generally be used to refer to the process of validating a single certificate. Certificate validation is composed of different verification processes, including check of certificate integrity, validity period, key usage and applicability according to certification policies. Last, but not least, the validation process must check the certificate status to ensure that it has not been revoked.

Check of the certificate integrity requires knowledge of the public-key of the issuing CA. This is a "trust" problem: if a RP does not directly trust the key of the issuer, a chain of CA certificates must be built up to a trusted CA. Each certificate in the chain is vouching for the identity of the previous CA in the chain.

Consequently, if the certificate to be validated is not issued by a trusted CA then another task, named "certificate path discovery" (CPD), needs to be performed. CPD tries to find a certificate sequence that leads to a trusted CA. This may require to construct several certificate chains before finding an acceptable one.

The process of constructing a certificate path rooted in a trusted CA is called *path construction*. The process of discovering a set of certificate paths leading to one or several trusted CAs is called *path discovery*.

Certificate path validation will be composed of validation of all certificates in the path, plus control that a number of constraints have been respected (e.g. naming and policy constraints).

One of the aims of this paper is to explain the complexity and the currently unsatisfactory implementation of certificate path discovery and certificate path validation. CPD and CPV together make up certificate path processing: this is a very important process because it represents the algorithmic foundation of trust in PKIs. If path processing is avoided or it is done in an incorrect manner, then the RP may be subject to serious attacks because it runs the risk of using invalid certificates. Nonetheless, current path processing implementations are rather unsatisfactory. Some commercial products implement the basic path validation algorithm described in [2], but several important features are missing. Examples are security-policy enforced configuration and maintenance

of specific-client validation parameters, automatic certificate path discovery and easy retrieval of relevant certificate revocation status. The actual certificate validation features of some commercial products and service providers are described in Sect. 8.

When discussing these problems, we will not restrict ourselves to the classical workstation model, that is a powerful computer with permanent high speed network access. PKIs are an important security support also for mobile and/or lightweight devices, such as laptops, PDAs and cellular phones. These devices are characterized by intermittent network access, low communication bandwidth and low computational power. For example, mobile and lightweight clients typically connect to the network for a short period of time (e.g. to download e-mail messages or to perform a transaction) and then disconnect and allow the user to locally perform off-line operations that may require certificate validation. Unfortunately, verification of cryptographic signatures and certificate validation require access to public keys and other related information, which could lead to heavy traffic loads. Similarly, messages cannot be encrypted without having the recipient's public key. Since public key are generally obtained from a network repository, the user may have to reconnect in order to prepare messages for sending or to verify the authenticity of a downloaded message.

Given this general framework, we'll try to identify the problems encountered by users mentioned above during certificate status discovery and certificate path construction and we'll give some recommendations for the steps to be done when a certificate path validation operation needs to be performed.

2 The Verisign-Microsoft Case

In mid-March 2001, VeriSign, Inc., a major certification authority, has warned Microsoft that on January 29 and 30, 2001, it erroneously issued two VeriSign Class 3 code-signing digital certificates¹ to an individual who fraudulently claimed to be a Microsoft employee. The identification assigned to both certificates is "Microsoft Corporation". The ability to sign executable content with keys that purport to belong to Microsoft would clearly be advantageous to an attacker who wishes to convince users to allow the content to run.

The attacker's overall goal would very probably be to convince other users to run an unsafe program, by using the digital signature to convince them that it is actually bona fide Microsoft software and therefore safe to run. Microsoft, like many other software developers, digitally signs its programs in order to assure that the programs are legitimate and have not been modified.

The attacker probably would choose to use an attack scenario designed to deliver the program to a large number of users, such as hosting the signed program on a web site or sending an HTML e-mail that would retrieve it from a web site. It's also likely that he would choose to package the program as either an ActiveX control or an Office document containing a signed macro, because doing so would allow him to initiate running the program automatically, as soon

¹ Class 3 certificates are the Verisign certificates with the highest level of assurance

as a user either visits the web page or opens the HTML mail.

From the attacker's perspective, the problem with simply sending a signed program as an attachment to a mail, or hosting it as a file on a web site, is that he would need to persuade the user to select the program and deliberately choose to run it. The attacker would want a method that minimizes the number of steps the user must take to run the program. Programs signed using these certificates would not be able to run automatically or bypass any normal security restrictions. The certificates are not trusted by default even though the two bogus certificates say they are Microsoft certificates. The users are guaranteed to see the warning dialogue the first time they encounter a program signed using either of these certificates, and will continue to see it unless he selects "Always trust content from Microsoft Corporation" in response to the warning dialogue. This is because trust is assigned on a per-certificate basis, not on a per-name basis. By viewing the certificate in such dialogues, users can easily recognize the certificates. Nevertheless the warning dialogue that appears before such programs could run would claim that Microsoft had digitally signed them. Clearly, this would be a significant aid in persuading even a security-conscious user to run the program.

After discovery of the error VeriSign immediately revoked the certificates, and inserted them in a list regularly issued and signed by VeriSign (technically speaking this is called Certificate Revocation List or CRL). However, because VeriSign's code-signing certificates do not contain the piece of data indicating the location from which the CRL can be obtained (CRL Distribution Point, or CDP), it is not possible for any certificate validation mechanism to locate and use the CRL. Moreover the CDP is readable only by some Microsoft products. Microsoft has developed an update that rectifies this problem. The update package installs on the local machine a CRL containing the two certificates and a revocation handler that consults the CRL on the local machine, rather than attempting to use the CDP mechanism.

As explained in [3] even though this cannot be tagged as a security vulnerability in Microsoft products and it is more a human error made by a third party, it clearly poses a serious risk to Microsoft's customers, which should immediately take actions. Because of the risk this issue poses, Microsoft has taken the unusual step of producing an update for every Windows operating system produced since 1995, regardless of whether it's normally supported or not.

This proves that current PKI-enabled products (like browsers) don't support certificate validation correctly while it is needed because even major organizations can make mistakes.

3 Architectures for Certificate Path Processing

A user of a security service requiring knowledge of a public key generally needs to obtain and validate a certificate containing the required public key. If the user doesn't already directly trust the certificate of interest or doesn't have a trusted copy of the certificate of the issuing CA then multiple certificates may

be needed. A list of certificates needed to allow a particular user to obtain the public key of an entity is known as *certificate path* or *certificate chain*.

Let $C = c_0, c_1, \dots, c_n$ be a chain of certificates, where c_n is the end-entity certificate of interest, c_0 is a self-signed trusted CA certificate, and c_k is signed by an intermediate CA c_{k-1} for all $k=1, \dots, n$. The goal of path validation is to verify the binding between an entity and a public key, as represented in the end-entity's certificate. The binding is limited by constraints, which are specified within the certificates in the path. The basic and policy constraints extensions allow the certificate path processing logic to automate the decision making process.

The verification is based on the public key of the *most-trusted CA* [2]. The *most-trusted CA* is a matter of policy: it could be a root CA in a hierarchical PKI, the CA that issued the relying party's own certificate(s), or any other CA in a network PKI. Basic Path Validation algorithm (further discussed in Sect. 7.1) assumes that most-trusted CA certificate is contained in a self-signed certificate.

On the other hand when a certificate handling system uses a certificate (e.g. for verifying a remote user's digital signature), that system not only checks the certificate signature and validity but also has to acquire a *suitably - recent* CRL for any certificate c_i in C and has to check that the certificate serial number is not on that CRL or it has to be able to contact a "special" server (technically speaking, an OCSP responder) to get a response indicating the status of the certificate. By definition, if any certificate c_i in C is revoked then C is an invalid chain. According to [2] the meaning of *suitably - recent* may vary with local policy, but it usually means the most recently issued CRL.

To study the certificate path validation process we defined the first architecture that is suited for on-line clients and applies mostly to current existing PKI-aware products. The PKI-enabled application and the validation module generally reside on the same system, or the PKI-enabled application and the validation module are incorporated in a single application. The certificates and revocation information are stored locally in a dedicated database or are downloaded from a repository. Each RP individually configures the validation rules. The general schema is presented in Fig. 1.

If certificate handling is going to be widely deployed in a variety of user applications and environments, the amount of processing an application has to perform before it can accept a certificate must be significantly reduced. If the certificate-enabled applications are running on a device with limited computing capabilities or with temporary network connections, then the above architecture has serious deficiencies. This led us to the idea of designing a new architecture, in which the client delegates partially (e.g. only path discovery) or totally (e.g. path discovery and path validation) the certificate path processing to a server. This second architecture is depicted in Fig. 2.

To understand the benefits of this architecture, we'll consider the steps which must be performed by the RP for path processing: firstly path construction, followed by certificate validation for each certificate in the path, along with veri-

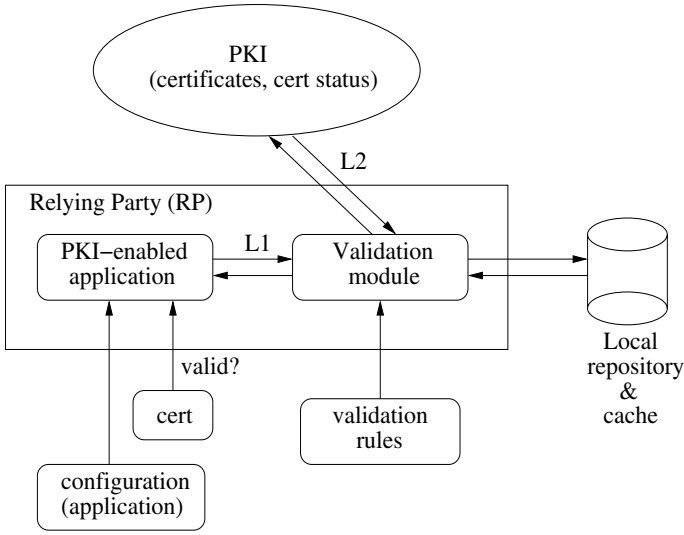


Fig. 1. Architecture of certificate validation for on-line powerful clients

fication of path constraints. Unlike the first architecture which was monolithic, the architecture of Fig. 2 is more flexible because it allows a RP to delegate some subtasks to another party.

For example the RP may delegate only the *path construction*. This may require a path discovery task. In this case, over the link L1 in Fig. 2 the RP sends to the validation server a certificate and some rules and receives back a path or a set of paths. The validation server behaves as a server whose task is path discovery.

Alternatively, or additionally, the RP may delegate *certificate validation* for each certificate in the path and *verification* of path constraints. This can be done in two alternative modes. In the first mode, the RP makes single queries for each certificate in the path to the validation server. In this case the task of the validation server is to perform just certificate validation: it receives from the RP a certificate and sends back a response about certificate' status. The RP will verify the path constraints by itself. In the second mode, the RP sends to the validation server an entire path and gets back the global validation status of the whole path. The validation server uses link L2 to retrieve certificates and certificate status information.

It is important to note that several servers could be used, some of them performing only certificate validation or path discovery, while others could support full path validation. Other servers could implement all the above services.

This architecture is useful for lightweight clients because it spares them to perform heavy calculations. It permits also to security administrators to avoid the burden of configuring one-by-one the clients, by allowing centralized administration of validation rules. Disconnected clients can benefit from the minimization of network traffic, because the data used in the validation process can be

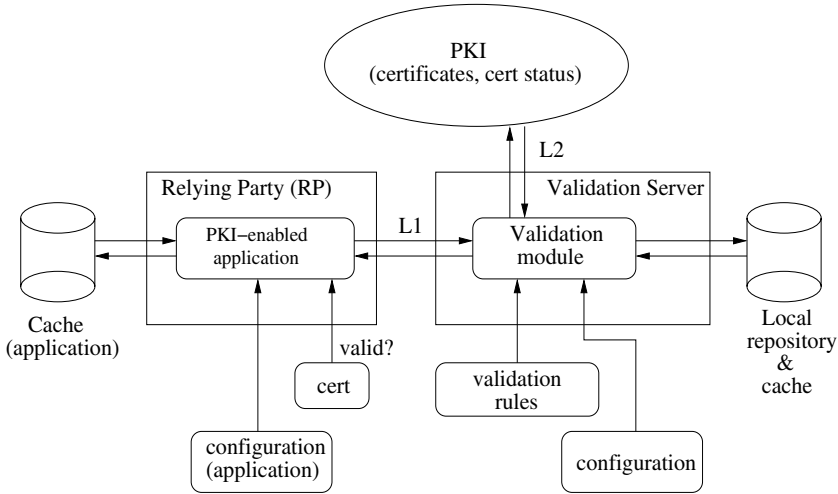


Fig. 2. Architecture of certificate validation for off-line, lightweight and mobile clients

cached. The architecture allows also storage of non-repudiation information on the validation server for later reference by the RPs.

4 General Requirements of Certificate Validation Process

In general, certificate path processing would need to be studied for two types of communities: closed or open environment.

Closed environment is a community administered in a centralized fashion and usually its users have certificates issued under one single trusted CA. The validation rules (policies) can be configured centrally on a server or the validation parameters can be easily configured locally on each client.

Open environment is a community of loosely coupled users, corresponding in practice to an open network with many parties of different trustworthiness and longevity. The security architecture for wide area network validation system spanning multiple administrative domains will require support for policy delegation and certificate distribution across the network.

The above type of classification is needed because different certificate revocation methods present advantages and disadvantages related to the network architecture, the management of validation policies is done differently in each of the above cases, as well as the certificate path discovery.

To support certificate validation, certificates need to respect a specific profile that provides information about access to certificate' status and location of the certificate of the CA that issued the certificate in question. Unfortunately, there is no agreement about where this information should be put inside the certificate. The following X.509 certificate extensions could be used:

- the *Issuer Alternative Name* extension can contain an URL, but the interpretation of this URL is demanded entirely to the specific application
- the *CRL Distribution Point* contains the location of CRLs, but it is within standardized practice to omit it
- the *Authority Info Access* extension can contain an URL indicating the location of the CA certificate and/or the location of an OCSP responder

If the certificate does not contain any of these extensions (as in the Verisign-Microsoft case) then the certificate's revocation status cannot be automatically checked, unless it is configured in some other way into the application.

Let us suppose now that an agreement has been reached about the certificate's profile to include the necessary data. However, other aspects about the complexity and architecture of the general infrastructure that should support such a complicated process and about the methods used for discovery of certificate paths and for the retrieval of certificate revocation status have to be studied.

At the moment there is no global infrastructure that could permit automatic retrieval of information required in certificate path processing. If the RP starts the certificate path look-up from the issuer DN field contained in the certificate to be validated he would have only an X.500 Distinguished Name (DN) but there is no worldwide X.500 Directory Service. Even if we consider the simpler LDAP Directory Service, there is not a global infrastructure.

One alternative solution to a global directory could be the complete deployment of the DNS security extensions proposed in the RFCs 2535-2541. The security extension defined in RFC 2538, namely the CERT resource record, is used specifically for storing certificates and their related CRLs. Since DNS is optimized for small data transfers (less than 512 bytes, which is obviously incompatible with the current size of certificates) a new extension was proposed. The EDNS0 extension (RFC 2671) allows clients to declare the maximum size of the UDP messages they are able to handle. In this way the overhead of a TCP connection can be avoided. With DNSSEC and EDNS0 extensions the DNS can be seen as a valid architecture to distribute certificates and revocation information.

When it comes to *certificate path discovery*, we can note that even if standards have defined certificate constraints to simplify automatic path discovery, potential technical flaws and security weaknesses still exist at the construction of the certificate paths in closed environment and even more in open environments. Some of the problems will be analyzed in Sect. 5.

For the *retrieval of certificate revocation status* several approaches have been proposed. Generally, it is very difficult to find revocation solutions that address both the timeliness and the performance requirements of all parties. At present, the most widely utilized revocation method is the use of CRLs maintained on repository servers. However, many implementations avoid CRLs because they may become too big and outdated to be relevant. Other efforts have been deployed so far for offering fresh and reasonable revocation information. An overview of several revocation methods is offered in Sect. 6.

All of the above considerations regarding the certificate's profile, the absence

of the worldwide certificate distribution architecture and techniques used for certificate path discovery and the retrieval of certificate's revocation status have to be taken into account when implementing a certificate validation service. With the help of the two architectures explained in Sect. 3 we'll try to identify further the parameters/requirements that should be taken into account in a certificate path processing service:

a. *Relying party computational capability.* The RP could be either computationally weak or strong. Questions like whether or not a RP has to be PKI-aware (that is, technologically enabled to verify signatures, to perform path validation and to be cognizant of validation parameters) or just Abstract Syntax Notation (ASN.1)-aware (that is capable of parsing ASN.1 objects) has been raised and are still under discussion in various groups of researchers. Significant benefits could accrue from using syntax other than ASN.1 (e.g. XML [4]) for the messages exchanged between the server and the client in the architecture of Fig. 2.

b. *Relying party network capability.* The certificate validation service should be available to a massive number of users, no matter if they are on-line or disconnected clients. The mobile users usually are located in wireless environment with narrow bandwidth channels and links relatively unreliable. Furthermore, off-line users could have also low computational capabilities as their devices are powered by batteries with limited lifetime. In this category of users with low network resources enter also the users connected to the Internet via dial-up connections.

c. *Interoperability issues.* Because at the moment each enterprise develops its own proprietary certificate validation solution, spending time and money to deploy a secure certificate path validation framework makes sense only if the framework is general and open enough to ensure validation service no matter what type of application is used (e-mail, commercial transaction, etc) and that is suitable for users that are permanently or intermittently connected to Internet. There exist currently several competing proposals for definition of a standard related to certificate path validation. This assumes not only the description of the bits on the wire but also the description of the syntax and processing to be done by each party involved.

d. *Transactional value (low security and high security applications).* Let's suppose that the user has already obtained the root certificates using secure out-of-band channels. Nevertheless, the discovery of revocation information implies costs proportional with its freshness and scalability. Usually the absolute necessity of finding the most recent revocation information depends on the value of the transaction performed. This is one of the validation parameters that must be input to the validation process and there must exist the possibility of users to easily configure these parameters, depending on the type of operation performed. In some cases it is preferable to configure the validation parameters centrally for a set of clients. It doesn't exist at the moment a Validation Policy System (VPS) that would provide mechanisms for configuring, discovering, accessing, managing and processing validation policy information.

e. Protocol requirements. A number of protocols like OCSP, HTTP and LDAP need to be supported by the RP in order to communicate with various servers to retrieve certificates and certificate status information.

5 Certificate Chains: Still a Problem to Construct Them

The simplest model of certificate chaining is the hierarchical chaining and is the only working model today. It mirrors most organization structures making it very popular for small-scale PKI deployment. This model is based on one or more certificate trees completely separated. The result is a set of linear certificate paths, from a trusted root certificate to the user certificate.

Usually each user (end-entity) trusts its own CA. If $user_1$ trusts CA_1 and $user_2$ trusts CA_2 and CA_1 and CA_2 are both elements in the same hierarchy then $user_1$ and $user_2$ trust the common top-root self-signed certificate CA_0 (see Fig. 3). Thus, they can easily discover and validate each other's certificate path.

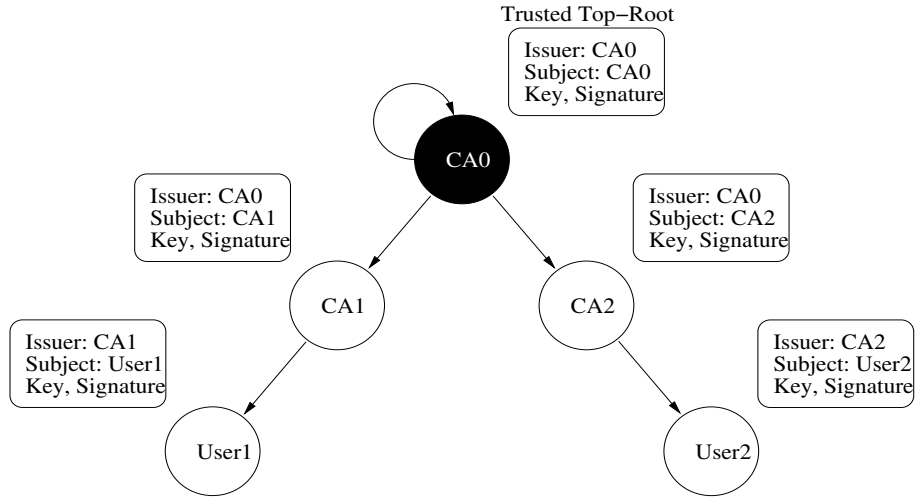


Fig. 3. Hierarchical chaining

This is the typical model deployed by most commercial CA today. For example, the two biggest banking international initiatives SET (from VISA and Mastercard) and Identrus are both using that model specific to closed environments. The model is based on a fixed and rigid hierarchy to deliver high value certificates in the context of payment services. The most important weakness of this model is that all the trust is based on the private key of the top-root CA. If the top-root key is compromised then the whole system is compromised. Consequently other out-of-band mechanisms have to be deployed in order to distribute top-root key to the end-users. The second important weakness of this

model appears when trying to extend in large-scale because it has to be somehow decided which organization to be the root of the roots.

The situation gets complicated when multiple hierarchies are involved. In this case, $user_1$ has to trust the top-root certificate of $user_2$ and vice-versa. The collection of trusted root certificate will usually be kept by each side in a list.

Nowadays in the Internet community the users receives the information needed for performing path processing, that is a set of self-signed root certificates and revocation data (usually CRLs) from repositories using X.500, LDAP, HTTP, FTP and SMTP protocols or get a reasonable set of generally trusted root certificates from files that come incorporated with several products (like web browsers). For example *mod ssl* [5] comes with a *ca-bundle.crt* file that builds from Netscape's *cert7.db* file. Thus, the responsibility of building and managing the list of trusted root certificates is left to product providers (Netscape, Microsoft, etc). This means that inclusion of top-root CA certificates is mainly based on business requirements than on trust requirements. Moreover we'll notice that people or security-administrator should further control/delete this list. This represents a serious problem due to the fact that some persons simply don't know how to manage this kind of information. Efficient path validation mechanisms depends on the certain knowledge of the public keys (and other information) about one or more trusted CAs (usually found in the form of "self-signed" certificates). The decision to trust a CA is an important decision as it ultimately determines the trust afforded a certificate.

Certificate path validation does not use a secret key, only public keys. In the certificate path processing there are used one or more *root* public keys. If an attacker can add his own public key to the list of root public keys, then he can issue his own certificates, which will be treated exactly like the legitimate certificates. They can even match legitimate certificates in every other field except that they would contain a public key of the attacker instead of the correct one. Assuming that in a highly secure application cannot be used any of the root certificates from Netscape database or other external pre-defined database, the best solution for the programmer is to contact the CAs directly and get a written statement for the root certificates he is interested in.

In addition to list management problem, where a key compromise or CA failure occurs for a trusted CA, the user will need to modify the information provided to the path validation software. Selection of too many trusted CAs will make the trusted CA information difficult to maintain. On the other hand, selection of only one trusted CA may limit users to a closed community of users until a global PKI emerges.

Two of the most widely used protocols, S/MIME [6] and SSL/TLS [7] are sending the certification paths as part of the protocol exchange. In general, the RP has at least one trusted root certificate obtained by an Out-Of-Band (OOB) method for the certification paths he receives and that certifies a high-value transaction. Some applications provide means for the entire (that means up to the root) certification paths to be transmitted as part of the protocol. Firstly we can notice that it doesn't help the RP just to hold the root keys in the trusted

root certificates. Even though the user has obtained the root certificates by secure OOB methods the storage and protection of the root certificates database still remains a problem. The only answer is to do all certificate validation on a computer system that is invulnerable to penetration by hostile code or to physical tampering. Secondly we'll note that the change in the revocation status of a certificate from the path requires that the input parameters of the path validation software to be modified.

We don't have to forget also that a chain of authorities is, by its nature, as strong as the weakest link because it relies on the correctness of every authority in the path. If any authority in the path incorrectly authenticates the next authority, then the user can be misled regarding the authentication of subsequent authorities in the path, including the public key of interest. A natural approach to increase assurance in the authentication of the name-to-public key binding is to use multiple paths [11]. However, the assurance provided by these multiple paths may be unclear, especially if they have several authorities in common or authorities that act in a correlated way. Thus, several authors [8,9,10] have proposed metrics to evaluate the confidence afforded by a set of paths. A metric takes as input several paths of authorities and returns a numeric value. A higher value indicates greater confidence in the name-to-public-key binding than those paths support.

6 Certificate Status Retrieval

Several methods have been proposed to allow the relying parties to retrieve the certificate status. This section gives a short overview of the most common revocation mechanisms, outlines their pros and cons (in terms for instance of timeliness, scalability, performance), and describes their different certificate status distribution model. All the available revocation mechanisms share the design goals of correctness, scalability, and availability: all verifiers must be able to correctly determine the state of a certificate within well-known time bounds; the costs for the determination of current revocation status of certificates should not exponentially grow with the increasing of the PKI user community size; replicated repository servers should be provided to ensure service availability and responsiveness.

Certificate Revocation List (CRL) is the most widely diffused solution for revocation. The CA signs and issues on a periodic basis a list identifying all the revoked certificates. CRLs may be distributed by exactly the same means as certificates, namely via untrusted channels and servers. The disadvantage is the increasing size of the CRL that leads to high repository-to-user communication costs. Thus, CRLs can introduce significant bandwidth and latency costs in large scale PKIs. Another disadvantage is that the time granularity of revocation is limited to the CRL validity period, thus the timeliness of revocation information is not guaranteed.

It's worth adding a security warning that clients who retrieve CRLs without verifying the server's identity run the risk of being sent an obsolete CRL.

Clearly an attacker cannot create a false CRL (without compromising the CA), but an attacker does have a window of opportunity to use a compromised key by denying access to the relying party to the latest CRL and providing access to a *still-valid-but-obsolete* CRL. Using the term *still-valid-but-obsolete* implies that CRLs have a validity period. Certificates have a validity period clearly specifying the start date and the expiry date of the certificate. CRLs instead have values for the date when a CRL is issued (called thisUpdate) and the date when the next CRL will surely be issued (called nextUpdate). But nothing prevents a CA from generating and posting a new CRL (let's call it *off-cycle* CRL) immediately a new revocation becomes known [11]. If an intruder deletes or impedes the access to an *off-cycle* CRL from an untrusted server and leaves the previous periodic CRL in its place, this cannot be detected with certainty by the RP.

Another important observation is that a CRL does not become invalid when the next CRL is issued, although some applications behave as though it does (see Sect. 8 for CRL processing in Netscape products). *CRL Validity* interpretation eliminates the ability of the client to decide, based on the value of information, how fresh its revocation information needs to be. If CRLs are issued every hour, a user might demand a CRL less than two hours old to authenticate a high value purchase transaction. If a CA issues CRLs every month, a user would rather prefer to be warned that, according to his application preferences, that CA's certificates shouldn't be used for high value purchases. This means that the user doesn't want the application to blindly treat them as just as fresh as certificates from CAs who issue CRLs daily or hourly.

Several CRL based alternatives attempt to overcome the scalability problem: windowed CRL [12], CRL distribution point (CDP) [13], delta CRLs [13] and Indirect CRLs are the most known.

One of the most used revocation method used by commercial companies is the CRL distribution point (CDP). A CDP is the location from which the RP can download the latest CRL and is usually listed as "CRL Distribution Points" in the details of the certificate. It is common to list multiple CDP's, using multiple access methods (LDAP, http, File protocols), to ensure that applications (that is, browsers, web servers) are always able to obtain the latest CRL.

In the Indirect CRL scheme, a different authority than the certificate issuer can issue a CRL. A CRL can be configured to contain revocation information from multiple certification authorities, so that the number of CRLs needed by end entities during certificate validation can be reduced. The overhead for obtaining multiple CRLs is reduced, and the recursive certificate verification for a certification path may be reduced. One way to deploy this scheme is by having the Indirect CRL issuer obtain CRLs from each individual CA and compile them into one CRL. The main performance gain in using indirect CRLs is that there is less overhead in requesting multiple CRLs from different sources. This can lower the request rate, as well as the average load. An important consideration for this scheme is that the indirect CRLs never should grow so large that the performance benefit of combining the CRLs is lost.

A delta CRL is a CRL that only provides information about certificates

whose status have changed since the issuance of a specific, previously issued CRL. A RP in need of more up-to-date information, that has already obtained a copy of the previously issued CRL, can download the latest delta-CRL instead of downloading the latest full CRL. Since delta-CRLs tend to be significantly smaller than full CRLs, this will tend to improve the load on the repository and the response times for the RP.

The other most widespread certificate status distribution mechanism is the On-line Certificate Status Protocol (OCSP) [14]. This protocol allows applications to determine the revocation status of a specified certificate by querying an online OCSP responder that provides to clients fresh information about certificate status. This mechanism can provide more timely revocation information than CRLs and seems also to scale well for large user communities. However, since certificate(s) status validation implies a specific client/server request to OCSP responders, the mechanism can overload the network and generate an intense traffic toward the OCSP responder. Thus, even if the cost of communication user - to - repository is lower compared to the traffic involved in transmitting a CRL, there still an intense communication toward the OCSP server. Moreover, the responses still has to be signed by the OCSP responder with a private key corresponding to the responder's certificate. From the security point of view authorities that have private information should be kept off-line. This leads to the conclusion that OCSP responders are very vulnerable to security attacks unless all the responses are pre-signed and the responder is kept off-line. The on-line protocol increases key management costs for public-key pairs and it requires also that client applications to generate and parse OCSP messages.

Another proposed mechanism for certificate status validation is called short lifetime certificates that assumes that certificates have a lifetime so short that they cannot be revoked, and are certainly valid until they expire. This solution avoids any burden of management of certificate status validation mechanisms, but significantly increases the number of issued certificates, and often requires an on-line CA that automatically issues short term certificates.

Further proposals for certificate status validation make use of Certificate Revocation Trees (CRT), authenticated dictionaries and authenticated search trees [15]. These approaches maintain, in tree data structures, a set of assertions about the status of issued certificates, but differently from CRLs they are aimed at proving whether a certain certificate is revoked without sending the complete list to significantly reduce the network load. The drawback of these schemes is that any changes in the set of revoked certificates, requires a re-computation (totally or partially) of the revocation trees. Furthermore, performance evaluation showed that the use of CRTs doesn't scale well compared to OCSP when the number of revoked certificates is big [16].

Others exploit trusted directories where a certificate is published only if it is valid, and it is removed from the directory as soon as it is revoked. The main difference of this solution with the others, is that it does not provide any form of non-repudiation: a user cannot prove to a third party that the certificate was effectively invalid at a certain time.

The issues in architecting a solution for certificate revocation notification to a validating party are very complex and require consideration of topics such as: the communication model, i.e., which class of users are communicating with which class of users, the trust model, the communication bandwidth available, size of the revocation response from the repository, processing load on the repository, processing load on the user workstation.

7 Standardization Efforts

7.1 Basic Path Validation

Basic Path Validation algorithm is described in [2] and begins with verifying that the signature affixed to the certificate is valid, i.e., that the hash value computed on the certificate contents matches the value that results from decrypting the signature field using the public component of the issuer.

To the path processing software have to be provided at least the following inputs: the certification path, a set of initial policy identifiers, the current date/time, the time, T , for which the validity of the path should be determined. The time T may be the current date/time, or some point in the past.

As we have already mentioned, the basic path validation software is still critical in executing all the steps due to the inherent security breaches in the construction of certification chains and due to the lack of quick acquisition methods of fresh revocation information. The algorithm specifies whether a particular certificate chain is valid but it doesn't describe how to discover certificate chains in the first place. Moreover, implementations of the basic path validation that omit revocation checking provide less assurance than those that support it. The availability and freshness of revocation information will affect the degree of assurance that should be placed in a certificate.

7.2 Remote Path Processing Services

Remote path processing provides two primary services and have appeared as a necessity to offer a solution to the problems related to the path validation and outlined in the above paragraphs. These services are generally called *delegated path construction for client-based path validation* (DPD) and *delegated path validation* (DPV). The basic idea in these proposals is to delegate either the path discovery or path validation task to a server. Obviously not all clients will require both services in all scenarios. Many clients require only delegated path construction (including path discovery) to help them perform their own path validation, while other clients have requirements for offloaded path validation and have no need for data acquisition. We have to mention here that the definition of such protocols is not closed and at the moment there exist some proposals (in draft format) for DPD [17] and DPV [18] as extension of OCSP, that is OCSPv2 [19]. The general certificate validation architecture is depicted in Fig. 4.

The delegated path discovery features of remote path processing are valuable for clients that do much of the PKI processing themselves and simply want

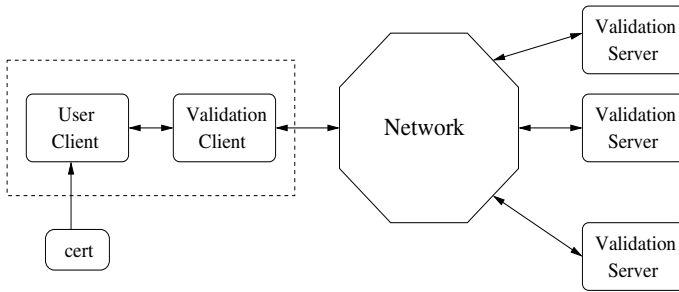


Fig. 4. General Certificate Validation Architecture

a server to collect information for them. The server need not even be trusted, because the client will ultimately perform path validation.

A client that performs path validation for itself may still benefit in several ways from using a server to acquire certificates, CRLs, and OCSP responses to aid it in the validation process. In this context, the client is relying on the server to interact with repositories to acquire the data that the client would otherwise have to acquire using LDAP, HTTP, and so on. Since these data items are digitally signed, the client need not trust the server to return the “right” data any more than the client would have to trust the repositories. There are several benefits to this approach; for example, a single query to a server can replace multiple queries to one or more directories and caching by the server can reduce latency. We can assume that the information maintained by the server is correct, as the cache consistency will not be analysed in this article.

Another benefit to the client system is that it doesn’t have to incorporate a diverse set of software to interact with various forms of repositories, perhaps via different protocols, nor to perform the graph processing necessary to discover paths.

The technological and geographic diversity of the sources of data motivates existence of service that enables relying-party software to acquire certification path data from an OCSP server rather than replicate the same functionality. The Delegated Path Discovery (DPD) extension to OCSP addresses these needs.

When we’ve tried to sketch the models for the study of certificate validation we’ve mentioned that a client that lacks the processing and/or communication capabilities to perform path discovery and path validation may require off-loading path validation to a server. We’ve noticed also that another motivation for offloading path validation is that it allows centralized management of validation policies in a consistent fashion across an enterprise. The Delegated Path Validation (DPV) extension to OCSP addresses this functionality for client/server systems as described in [18].

In this case, there must be one (or more) fully-trusted server(s) to which the client can delegate path validation services. Such a server can take direction from the client about how path validation should be done (such as which roots are to be trusted), and the server returns a valid/invalid response and can even pro-

vide to the client proof (e.g., certificates and CRLs or OCSP responses or a hash of the evidence) of how the server validated the target certificate. Even clients that are able to do their own path validation might instead rely on a trusted server to do path validation if the client is in an environment where centralized management of trusted roots or centralized management of non-standard policy processing is needed for some applications.

There is ongoing work within the Internet Engineering Task Force (IETF) to address this issue: the Simple Certificate Validation Protocol (SCVP) [20] is being proposed as an alternative protocol, while DPD and DPV have been proposed to be defined independently [21] and not as extensions to OCSP.

Even though these proposals seem attractive in terms of reducing the complexity associated with retrieving and processing the information by the RP, there exists some drawbacks. One disadvantage is that if a trusted server is compromised then all relying parties depending on it will be affected. The RP will have to decide also how often to consult the trusted server and how long to cache and use the previous responses. Research is foreseen also about the level of complexity of the back-end infrastructure needed to support this approach. Bandwidth considerations, validation policies management and issues associated with caching replies need also to be explored further.

8 Commercial Products and Services

8.1 Network Security Services (NSS)

Network Security Services (NSS) [22] is the security library developed by the Mozilla organization [23] and used by Netscape products. The Netscape products (i.e. Netscape Navigator and Messenger) use in the validation process the information stored in the permanent Berkeley database *cert7.db* and a temporary database located in memory cache. For Netscape browsers there exist a list of trusted root certificates placed in the *Signer Info* page, in the *Certificates* tab under *Signers*. Thus, the addition/deletion of other trusted root certificates is done independently by users or system administrators. NSS doesn't present a mechanism for automatically fetching the CRLs. Netscape Communicator has an User Interface to help users fetch CRL's. The first time a client downloads a CRL a new button *View/Edit CRL's* is placed in the *Signer Info* page, in the *Certificates* tab under *Signers*. The CRL's locally stored in the *cert7.db* database are used in the validation of certificates. The main drawback is that NSS treats "last update/next update" fields in a CRL as expiration fields. If a loaded CRL "expires" then the user has to remember to download a fresher one otherwise NSS will reject all certificates certified by the CA issuing that CRL. The only operation that can be done in this case (unless of course downloading the last CRL) is to delete CRL and continue processing without making revocation checking. Another important drawback is that NSS only supports simple CRLs v2 and does not understand or parse any extensions. An the last point to be underlined here is that CRL processing can lead to incorrect results when time passes. For example, if a RP receives signed email from someone, and then later someone's

key is compromised, that previously valid email will show up as invalid. Likewise if a RP receives email signed with a compromised certificate, and later the certificate expires and is removed from later CRL lists, that email will suddenly show up as valid.

8.2 Personal Security Manager (PSM)

The Personal Security Manager is developed by the Mozilla organisation and is meant to provide cryptographic support for client applications. This includes setting up an SSL connection, object encryption/decryption, object signing, signature verification, certificate management (including issuance and revocation), and other usual PKI functions. The PSM is built on the NSS [22] library sources. The PSM version 1.3 is already integrated in Netscape 6 Public Release 3. The revocation methods supported by PSM include static (CRL-checking) and dynamic (OCSP) validation of certificates status. There exist three modes for NSS with respect to OCSP:

- *OCSP checking off.* Authority Info Access extension is ignored and the certificates are processed without revocation checking
- *OCSP checking on.* If the certificate has the Authority Info Access extension containing the URL of an OCSP responder, this responder is contacted
- *Default OCSP responder set.* All the certificates are verified against a default OCSP responder

When set, the OCSP checking of the certificate status has precedence over CRL checking.

8.3 Microsoft

As we noticed previously, Mozilla and Netscape products use a Berkeley database to store the digital certificates. However Microsoft uses its own mechanism to store the digital certificates. There is in fact more than one store generically called “MY”, “CA”, “ROOT”, and “AddressBook”, in Windows. It doesn’t exist at the moment a unified database to store digital certificates. Thus the users need to import the same digital certificate(s) twice when they have more than one browser on their computer.

Internet Explorer 5 can be set to check for the revocation status of certificates on the *Advanced tab* in the Internet Explorer *Properties* dialog box. There are two checkboxes called “Check for publisher’s certificate revocation” and “Check for server certificate revocation (requires restart)”. Once these checkboxes are checked the IE browser retrieves a CRL when presented with a certificate for which it currently does not have the associated CRL in its cache or if the cached CRL has passed its effective date.

The main drawback is that it is not possible to apply these settings to verify certificates only by certain CAs. Another drawback is that the browser will automatically try to connect to download the CRL when these options are set

and when a certificate has been received (for example in a signed e-mail) from another party and the cached CRL (if present) has expired. The user isn't even notified of the CRL downloading in any way. If the RP is scarce in network resources he will practically not be able to perform CRL-checking because he would have to download large CRLs.

Recent versions of Internet Information Services (IIS) support real-time CRL checking to ensure that the client certificates authenticating to a web site are valid (not revoked). By default, certificate revocation checking is disabled in IIS v4.0 due to the time involved in performing the check, especially if the CRL happens to be on another server across the Internet. However, IIS v4.0 supports CRL checking even if the modality is somehow cumbersome because it requires for the user/web administrator the operation (sometimes dangerous) of editing the registries.

In Microsoft IIS v5.0 and Microsoft Certificate Services v2.0, CRL checking is enabled when the *CertCheckMode* metabase property is set to 0, and the web site or web page requires client certificates.

However, even if you have CRL checking enabled and the Web site or Web page requires client certificates, the client may still successfully authenticate to the IIS computer even when the client certificate has been revoked. The cause of this behaviour is that the IIS computer does not process any changes or revocations to a certificate until a new CRL is published. The default time for this is one week in Microsoft Certificate Services v2.0. The solution is to change the *Publication Interval* setting for the CRL from the default time of one week to a shorter duration. An HTTP error message can also occur if IIS (v 5.0) is unable to contact the server that stores the CRL when checking for revoked certificates. If a CRL server is unreachable, the certificates are presumed to be revoked.

8.4 VeriSign

VeriSign has developed an OnSite Certificate Validation Module [24] to enforce automated CRL checking in control applications. The Validation Engine can be configured to locate CRLs (and the CA certificates) using either HTTP or LDAP. Local, distributed and delegated CRL hosting is supported. The CRLs for the VeriSign's CAs can be found at <http://crl.Verisign.com/>. Moreover VeriSign supports the OCSP protocol by providing this service - over HTTP connections - at the URL <http://ocsp.Verisign.com/ocsp/status>.

8.5 Thawte

Thawte Co. supports CRL Distribution Point (CDP) as revocation method. Their CDP can be accessed via http at the following URL, under the *Certificate Revocation Lists* link: <http://thawte.com/developers/contents.html>. Thawte's CRLs are updated every few hours, or upon request to Thawte's network operations center.

8.6 Entrust

Entrust/PKI solution supports CDP method for validating certificates within Entrust enterprise deployments. Entrust and ValiCert (whose services are presented in the next paragraph) have recently entered into a strategic relationship whereby ValiCert, with its Validation Authority (VA) solution, is providing PKI interoperability and validation services to Entrust customers. Consequently Entrust and ValiCert have worked to provide OCSP validation capability required especially by financial service industry.

8.7 ValiCert

ValiCert has been a leading commercial provider of certificate revocation standards and solutions. Valicert supplies the Validation Authority (VA) solution for quickly and efficiently validating any digital certificate from any CA, for many applications. ValiCert does not issue digital certificates and is not a CA. It is a company exclusively oriented on VA solutions, hence providing a large range of validation methods (e.g., CRL/CDP, CRT, OCSP, SCVP, etc.).

9 Conclusions

This work represents first of all an analysis of the current weaknesses and problems encountered when designing and implementing a certificate validation service and it has involved deep study of the concepts related to public key certificate handling and certificate revocation mechanisms. Even if at the first sight the certificate validation could seem an easy matter, the analysis we've done so far clearly indicates that the models actually deployed are much more business oriented than trust oriented. Because this service will be addressed to a large number of clients with different processing and network capabilities and transactional requirements it becomes more and more clear that the complexity of certificate path processing must clearly be moved from end entity side to a third party authority.

References

- [1] M. K. Reiter, S. G. Stubblebine, *Towards Acceptable Metrics of Authentication*, Proceedings of IEEE Symposium on Security and Privacy, Oakland, CA, May 1997, pp. 10-20.
- [2] R. Housley, W. Ford, W. Polk, D. Solo, *Internet X.509 Public Key Infrastructure Certificate and CRL Profile*, RFC 2459, IETF, 1999
- [3] *Erroneous VeriSign-Issued Digital Certificates Pose Spoofing Hazard*, Microsoft Security Bulletin MS01-017, available at <http://www.microsoft.com/TechNet/security/bulletin/MS01-017.asp>
- [4] T. Bray, J. Paoli, C.M. Sperberg-McQueen, *Extensible Markup Language (XML) 1.0, W3C Recommendation*, 10-February-1998, available at <http://www.w3.org/TR/1998/REC-xml-19980210>

- [5] <http://www.modssl.org>
- [6] B. Ramsdell, *S/MIME Version 3 Message Specification*, RFC 2633, IETF, 1999
- [7] T. Dierks, C. Allen, *The TLS Protocol Version 1.0*, RFC 2246, IETF, 1999
- [8] T. Beth, M. Borcherding, B. Klein, *Valuation of trust in open networks*, European Symposium on Research in Computer Security, ESORICS 1994, vol. 875 of Lecture Notes in Computer Science, Springer-Verlag, 1994, pp. 3-18
- [9] U. Maurer, *Modelling a public-key infrastructure*, European Symposium on Research in Computer Security, ESORICS 1996, vol. 1146 of Lecture Notes in Computer Science, Springer-Verlag, 1996, pp. 325-350
- [10] M. K. Reiter, S. G. Stubblebine, *Path independence for authentication in large-scale systems*, Proceedings of ACM Conference on Computer and Communications Security, Zurich, Switzerland, April, 1997, pp. 57-66
- [11] W. Ford, M. S. Baum, *Secure Electronic Commerce*, Prentice Hall, 1997
- [12] P. McDaniel, S. Jamin, *Windowed Key Revocation in Public Key Infrastructures*, Tech. Rep. CSE-TR-376-98, EECS(Univ. of Michigan), 1998
- [13] ITU-T Recommendation X.509-ISO/IEC 9594-8, 1995
- [14] M. Myers, R. Ankney, A. Malpani, S. Galperin, C. Adams, *X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP*, RFC 2560, IETF, 1999
- [15] M. Naor, K. Nissim, *Certificate Revocation and Certificate Update*, IEEE Journal on selected areas in communications, Vol. 18, No. 4, 2000, pp. 561-570
- [16] H. Kikuchi, K. Abe, S. Nakanishi, *Performance evaluation of public-key certificate revocation system with balanced hash tree*, Proceedings of International Workshops on Parallel Processing, Wakamatsu, Japan, 1999, pp. 204 -209
- [17] M. Myers, S. Farrell, C. Adams, *Delegated Path Discovery with OCSP*, *draft-ietf-pkix-ocsp-path-00.txt*, IETF Internet Draft, September 2000
- [18] M. Myers, C. Adams, S. Farrell, *Delegated Path Validation*, *draft-ietf-pkix-ocsp-valid-00.txt*, IETF Internet Draft, August 2000
- [19] M. Myers, R. Ankney, C. Adams, S. Farrell, C. Covey, *Online Certificate Status Protocol, version 2*, *draft-ietf-pkix-ocspv2-02.txt*, IETF Internet Draft, March 2001
- [20] A. Malpani, P. Hoffman, R. Housley, *Simple Certificate Validation Protocol (SCVP)*, *draft-ietf-pkix-scvp-05.txt*, IETF Internet Draft, June 2001
- [21] D. Pinkas, *Delegated Path Validation and Delegated Path Discovery Protocols*, *draft-ietf-pkix-dpv-dpd-00.txt*, IETF Internet Draft, July 2001
- [22] <http://www.mozilla.org/projects/security/pki/nss/intro.html>
- [23] <http://www.mozilla.org/>
- [24] <http://www.VeriSign.com/onsite/datasheets/validation/validation.html>

Liability of Certification Authorities: A Juridical Point of View

Apol·lònia Martínez-Nadal and Josep L. Ferrer-Gomila

Universitat de les Illes Balears, Carretera de Valldemossa km. 7.5, Palma de Mallorca,
07071, Spain
{dpramn0, dijjfg}@uib.es

Abstract. Liability is an essential but a non-resolved question of commercial and legal development of certification entities. The issuing, distribution and use of a certificate, together with an eventual revocation or suspension of same up to its expiration date, generate relationships between implicated parties (basically, the provider of certification services, the subscriber and the user of the certificate) which set up the need to limit and clarify respective rights, obligations and eventual liabilities of each party. We analyze liability of certification authorities from a juridical point of view; the study is centered mainly in the content of the European Directive and the Spanish Law on electronic signatures, but we also refer to other laws (such as Utah Digital Signature Law, and Italian law on electronic signatures). We conclude criticizing legal rules on liability because they are incomplete and excessive, without taking into proper account the necessary balance of all involved parties.

1 Liability of Certification Authorities: General Considerations

Liability is an essential but a non-resolved question in these initial stages of commercial and legal development of certification entities. The issuing, distribution and use of a certificate, together with an eventual revocation or suspension of same up to its expiration date, generate relationships (of not always a clear nature) between diverse implicated parties (basically, the provider of certification services, the subscriber and the user of the certificate) which set up the need to limit and clarify respective rights, obligations and eventual liabilities of each party. Because the existence of uncertainties on liability of certification authorities could seriously affect progress of these entities.

The spread of commercial certification authorities depends in large measure on the degree of risk associated with business activity, which is potentially quite high with the current general rules of liability. When a certificate is issued, it can be sent together with any of various electronic messages in diverse operations which for the most part would be unknown by the certifier. Unlike credit cards system (in that, every time the card is used for an operation above the allowed credit limit, the issuer of the card is able to calculate the possible liability), in the case of the certificates, the certification entities do not have the possibility to authorize each operation individually, since normally they will not know of the specific transactions undertaken by the certificate subscribers. Given the indefinite number of electronic

operations made using the same certificate, it is difficult in these circumstances to fix a limit to liability (this also could impair negotiations by the certifiers to insure against liabilities). Hence the need to establish and delineate very clearly the rules and conditions of liability derived from the issuing and use of certificates, taking into account the interests of all parties in the electronic transaction (not only the certifier but also, e.g., a consumer subscriber or user of a certificate).

In this sense, most legislative initiatives on electronic signatures and certification authorities approach the delicate question of liability of these entities. In this paper we analyze the question of liability of certification authorities from a juridical point of view; the study is centered mainly in the content of the European Directive on electronic signatures (1999) and the Spanish Law on electronic signatures (1999), but we also refer to other laws (such as Utah Digital Signature Law, the first law in the world on this matter, approved in 1995 and modified later; and Italian law on electronic signatures, approved in 1997).

2 Situations of Liability of the Certification Authority

There are two main situations of liability for the certification authorities: a general liability once a certificate has been issued, and a specific liability in case of a revocation of a certificate.

2.1 Liability of the Certification Authority Once Certificate Has Been Issued

It has to be stated from the beginning that, unlike other legislations (see, for example, Utah Digital Signature Law) the European Directive does not regulate the rights and obligations of the parties participating in the certification system, and neither does Spanish law. And they don't rule correctly the liability derived from the issuing, distribution, use, revocation or suspension, and expiration of a certificate for the provider of certification services or for the subscriber of the certificate.

The Directive, and the Spanish law, obviate a complete and detailed regulation of the liability assumed by the different subjects of the system of certificates and only rule (art. 6, and art. 14, respectively), in a generic way, liability of the provider of certification services to any person who reasonably trusts in the certificate. With this general prevision is resolved the debated problem of the relationship between the third party that uses a certificate in order to verify an electronic signature and the certifying entity; a relationship which is, in principle, extra-contractual but which could be considered as contractual and legal according to the circumstances; in any case, with the previsions of the Directive and Spanish Law, and in general, according to all other legislations, this relationship causes a legal liability; certification authorities can not exonerate themselves from this liability to the third user of the certificate.

In fact, this first section of article 6 of European Directive establishes that "As a minimum, Member States shall ensure that, by issuing a certificate as a qualified certificate to the public or by guaranteeing such a certificate to the public, a certification service provider will be held liable for damage caused to any entity or

legal or natural person who reasonably relies on that certificate" in the following ways:

1. as regards the accuracy at the time of issue of all information contained in the qualified certificate.
2. for assurance that at the time of issue of the certificate, the signatory identified in the qualified certificate held the signature creation data (in the case of asymmetric cryptography, the private key) corresponding to the signature verification data (the public key) given or identified in the certificate; and
3. for assurance that the signature-creation data and the signature-verification data can be used in a complementary manner in cases where the certification service provider generates them both.

In the same way, art. 14.1 of Spanish law establishes the general rule of liability for certification entities, as follows: "Lenders of certifications services will be held liable for damages caused to any person in the exercise of the activity of certification ...".

These provisions of article 6.1 of European Directive, and art. 14.1 of Spanish Law, will engender a wide spectrum of liability for the certifying entity, and raises the question of error. In the case of a false certificate, how exactly will the certification authority be held liable? Only in the case of negligence of his employees (for example, if the identity of the subscriber were not examined)? Or also in the case of an error produced in spite of having acted diligently? (That is, if the authority is responsible for a falsification that was practically perfect and difficult to detect.)

2.2 Liability in Case of Revocation

A second and special case of liability is the liability generated in case of revocation of a certificate. The revocation of a certificate is normally caused by a compromise of the private key (e.g., because of loss or robbery of that key). That compromise demands the anticipated ending of the operational period of the certificate, to avoid non authorized uses of the private key by a third person (different to the legitimate holder).

In order to make possible the revocation, it's necessary, from a technical point of view, the establishment of mechanisms of revocation; the Directive only states that certification service provider must "ensure the operation of a prompt and secure directory and secure and immediate revocation service" (Annex II, b)). This requirement, also established in Spanish Law (art. 12,c), is too wide and generic, and, perhaps, it could produce the effect of exclude non-immediate methods of revocation such as for example, periodically updated revocation lists. And this requirement must be fulfilled not by every certification service provider but only providers issuing an special kind of certificate: the so-called "qualified certificates".

From a juridical point of view, in case of revocation of a certificate, it's necessary to establish and delineate very clearly the rules and conditions of liability derived from the revocation, taking into account the interests of all parties (not only the certifier but also, e.g., a consumer subscriber or user of a certificate). However, the directive, art. 6.2, only provides that: "As a minimum Member States shall ensure that a certification service provider who has issued a certificate as a qualified certificate to the public is liable for damage caused to any person who reasonably relies on the certificate for failure to register revocation of the certificate unless the certification service provider proves that he has not acted negligently". In the same way, art. 9.2

and 3 of Spanish Law states that a certification entity must publicize the extinction of a certificate, and that this entity will be held liable for damage caused to the subscriber or third users of the certificate due to delay in the publication of revocation.

The existence of these provisions in European Directive and Spanish Law, is positive. But the content of this provision is partial and incomplete. It doesn't consider properly the distribution and assignation of liability among the different parties (certifying entity, subscriber and third user) during the period of the revocation (from, e.g., the date of compromise of the private key to the effective publication of the revocation of that certificate corresponding to that private key). There is not offered a clear option between the different technically possible ways to publicize the revocations (certificate revocation lists, broadcast revocation lists or immediate revocation) which can suppose different distributions of liability. In consequence, there are yet unresolved questions, not approached (or only partially) by the legislator. For example, what would happen if, as the consequence of the loss of a private key, that key were used for an illegitimate purpose by another party? What would happen if the subscriber of the certificate had requested a revocation but the certifying entity hadn't in fact cancelled the certificate out of negligence, or if it were still checking the request? What would happen if the decision to revoke a certificate had been taken but wasn't yet known to the third relying party by reason of the time taken for publication of the information by the system (for example, periodically updated revocation lists)? And what would happen if the provider of the certification service revoked a certificate accidentally?

Next we analyze the problem of liability in case of revocation that is ruled only in a partial way in the different legislations. The regulation is partial and insufficient since every temporary period and all the implied people are not contemplated: only the publication phase, the effects of the publication in front of third parties and the possible delays in the publication on the part of the certification authority are regulated. Attending to the lack of a complete legal solution, the decisive approach should be the distribution of the risk among the implied parties in a reasonable way, so that the biggest risk will be assumed by the party with better capacity of control of that risk; or by the party that, according to his/her situation, will be the most appropriate to assume it, for example, because he/she has not created the risk but the situation that potentially generates it. With the final objective of getting an equal distribution of liabilities among the different parties, so that the liability will be assumed by the most suitable person. Keeping in mind that this person, as we will see, can vary in function of the different phases of the temporary process of revocation.

First Period. *The compromise of the private key is produced but it has not been notified to the certification authority (even is possible that the subscriber of the certificate doesn't know the compromise of the key).* In this first period, the security of the private key has been compromised and it is used by a third party, to supplant the holder of the certificate that doesn't still know the compromise, or although he knows it, he has not still notified it to the certification entity requesting the revocation of the certificate (either because of the brief time that, in spite of his extreme velocity, inevitably will lapse among the moment of knowledge and the moment of notification, either because of the widest time that will lapse if he doesn't act with diligence). The question is who assumes the liability for the non-authorized uses of the key during this period. Neither the European Directive, nor the Spanish law,

establish an express solution for this question. But it is possible, as we will see, to deduce some approaches of solution.

The *user of the certificate* does not know, and he is unable of knowing, the compromise of the key: the revocation not only has not been published, but rather it has not even been decided by the certification entity and even it is possible that the compromise of the key is not still known by the subscriber of the certificate. It doesn't seem reasonable to oppose and allege in front of third users of certificate a compromise that has not still caused the revocation of the corresponding certificate. And this doctrinal principle finds legal support in art. 9.2 of Spanish Law that states, as general rule, that the extinction of the effectiveness of a certificate will produce effects from the date in that the certification entity has exact knowledge of anyone of the decisive facts of this extinction and publish it in the register of certificates that should maintain obligatorily according to art. 11.e. Therefore, the simple compromise doesn't have effects in front of third parties, it is necessary not only the communication to the certification entity but the publication of the revocation.

Anyway, we have to consider the limit of "bad faith", express or implicitly admitted in different legislations approved or in elaboration, and that anyway could be applied attending to the general principle of "good faith" present in all legislations (principle that demands a fair play of the parties, without cheating or trying to achieve unjust advantage). So that if this third party had had knowledge of the compromise and of the intention of revoking through other ways (for example, because the holder of the certificate, once discovered the loss of the key, in a simultaneous, and even previous way to the remission of the revocation petition to the certification entity, communicates that circumstance to a certain user, noticing him that he should stop to trust the certificate, because, e.g., he (the subscriber) has lost the control of the private key corresponding to the certified public key), he (the third party) could not allege ignorance, and the presumption of ignorance would be eliminated. In this sense, art. 10.5 of the Italian regulation on digital signatures establish that the use of a digital signature created by a revoked, expired or suspended key is equal to lack of signature; the revocation or the suspension have effect, according to this art. 10.5, from the moment of the publication, except if the person who requests the revocation, or the suspension, demonstrate that this fact was already known by all the interested parties.

The *certification entity*, in this first period, does not know the compromise, because the subscriber has not request the revocation. And, as art. 9.3 of Spanish Law establishes (and so does art. 6.2 of European Directive), the certification entity has the obligation of publishing the extinction of effectiveness of the certificate and he will be liable of possible damages caused by delays in the publication (damages caused for failure to register revocation, according to art. 6.2 of European Directive). But, since the revocation has not been requested, it's not possible to refer to publication or delay in the publication. Unless the certifier entity has knowledge of the compromise, or of any other cause of anticipated extinction of the certificate, by other means, different from the subscriber's notification.

Finally, regarding to the *subscriber* of the certificate, it can happen that he had knowledge of the compromise before but also after the non-authorized use of the compromised key. And it can also happen that the revocation cause (e.g., the loss or robbery of the private key) is produced due to his own negligence or in spite of his extreme diligence. In principle, and since he has the obligation of keeping secure the private key, it can be reasonable that the subscriber of the certificate assumes the risk for non-authorized uses of the private key during this period, because this non-

authorized uses are consequence of the non-fulfillment of his custody obligation. It would be an approach similar to the solution of the distribution of risks in case of use of credit cards; in that case, legislation and courts generally admit that the holder of the lost or robbed card is liable, in these first moments, and till to the notification to the issuer, for the possible non-authorized uses of the card. And it seems to be the solution that can be implicitly deduced of the regulation of Spanish law because if the content of this law excludes, as we have seen, to the third user and the certification entity, this liability must be assumed finally by the holder of the certificate. And it seems the silence of the different legislations on electronic signatures leads also to this solution: the subscriber of the certificate should be liable for non-authorized uses of the private key, in this period, and, as we shall see below, till the notification to the certification entity and the publication of the revocation.

In a similar way to the system of credit cards, this liability of the subscriber seems to be strict and objective, with independence of his diligent or negligent actions. And it seems the silence of the different legislations on electronic signatures leads again to this solution: because if these regulations exclude, as we have seen, the liability of the third user and the certification entity in this first period, this liability finally must be assumed by the subscriber, without any kind of exoneration or limitation in case of diligence. This solution can produce excessively onerous results, especially in case the holder of the certificate is a consumer, because he always would be liable of non-authorized uses of the key, with independence of the compromise was produced because of his negligent acts or in spite of his diligent acts. These excessive results could be palliated establishing quantitative limitations of liability, or appealing to the insurance of the risk assumed, or establishing the possibility of being liberated of that responsibility carrying out certain performances (requesting the revocation of the card, and, in our case, of the certificate). These palliative elements are actually applied in the system of credit cards (e.g., the European Recommendation on electronic payments, approved in 1997, establish a quantitative limit of 150 euros) and we consider they are not only equally applicable to the system of certificates but even specially necessary in this second system, given the limitless of the risk assumed, under those conditions, by the subscriber of the certificate.

On the other hand, the negligent or diligent actions of the holder of the key could be important for the application of these palliative; so that, in case of the compromise of the private key was produced due to serious negligence of its holder, the limitation of responsibility and the covering insurance of the risk offered by the certification entity to their clients subscribers of certificates would not be applied, and these instruments only would benefit to those holders of certificates that had not been negligent. And this negligence could also be kept in mind by the certification entity that could refuse the application for the issuing of a new certificate to that applicant who had been the holder of a previous certificate that must be revoked due to its holder's negligence.

Second and Third Period. *The compromise of the private key has been notified to certification authority, but the decision of revocation has not still been adopted (second period), or, once adopted the decision, the revocation has not still been published (third period).* In these second and third periods, once the subscriber has had knowledge of the compromise of the private key, he requests the revocation of the corresponding certificate but a third party uses the compromised key, supplanting the subscriber, during the period among the request of revocation by the subscriber, and

the adoption of the decision of revocation and the inclusion in the corresponding list by the certification entity.

In order to distribute the existent risk in this period, *the user of the certificate*, as in the previous period, is unable to know the compromise of the key. So, with the same arguments and considerations, and the same legal approaches exposed in the previous period, it seems reasonable to establish that we can not allege to the user this revocation, because it is a fact that he has not had the possibility of knowing it. This approach seems confirmed by the principle settled down in art. 9.2 of Spanish law, and in most legislations on electronic signatures, that links the effectiveness of the extinction of a certificate to the publication in the register of certificates of the certification authority.

Therefore, it is necessary to distribute the risk between *the holder of the certificate* and *the certification entity*. The holder, as we have seen in the previous period, assumes, in the first moments, the liability for non-authorized uses of the private key, in a similar way to the holder of a credit card. Nevertheless, the same as in the credit cards system, the holder of the certificate should have the possibility of liberating himself of that liability communicating the compromise to the certification entity and requesting the revocation of the certificate. It would be a burden, a special kind of duty of the subscriber: an action that he must carry out if he wants to be liberated of that initial liability, transferring it to the certification entity; otherwise, the subscriber should assume the consequences that derived of his inactivity, it's to say, he will continue being responsible for the possible non-authorized uses of the private key. The problem is that, between the communication and the effective revocation (and its publication), a temporary period (more or less wide) will always exist, and it causes the uncertainty of who will be liable for non-authorized uses during this period. It is, in fact, the problem of determining the exact moment since the subscriber liberates himself of his initial liability and transfers it to the certification entity. Because, once received the corresponding request of revocation, it is indispensable that the certification entity confirms this request, so an inevitably interval will lapse between the reception of the request and the effective decision of revocation, and between this decision and the effective publication of the revocation.

And the possible solutions for this problem are diverse: first, we can consider liable to the holder that would not be liberated until the publication of the revocation; second, we can consider more reasonable attribute the liability to the certification entity that will assume the biggest risk for wrong use of the private key during this period; or, third, it is necessary to consider as key element for solution if the certification entity acted with the necessary velocity, and this question would depend on the trade practices and usages, but it is difficult to answer it when there are not usages established and the technology varies quickly.

Most of legislative initiatives on electronic signatures and certificates don't solve in a correct way the topic of the distribution of risks during the process of revocation of the certificate. In some cases they don't approach the question, and if they make it they seem to be limited to the distribution of risks in front of third parties (that appears connected to the effective publication of the revocation), but not to the question of the transmission of the risk from the subscriber to the certification entity that practically has not been contemplated (except in the Law of Utah, and with the effects that we will analyze next).

Utah Digital Signature, in art. 307.2 (*time to effect requested revocation*) establishes that a certification authority with license will confirm the request of

revocation and it will revoke the certificate in the next working day after receiving the subscriber's written request and a reasonably enough evidence to confirm the identity and the person's applicant representation. According to art. 307.6, two days after the subscriber requests the revocation in writing and provide to the certification authority reasonably enough information to confirm the request, and pay any rate required contractually, the effects of the revocation will take place (it is also necessary to understand these are effects in front of third users).

Art. 307.2 establishes, therefore, a term of one day so that the certification authority proceeds to revoke a certificate. And art. 307.6 seems to face the problem of the lack of diligence and velocity of the certification authority, settling down that, after two days, the effects of the revocation will take place anyway. In this way, the regulation of Utah seems to be more concrete and exact than others, like the Italian Law which has temporary indeterminations (because, for example, requires simply a temporary correct and immediate publication, without more concrete requirements). Nevertheless, according to the content of these precepts of the Utah Law, the beginning of the term of one day (or two days) is not from the reception of the request but from the confirmation of the request by the certification authority. So that, if the certification authority doesn't receive reasonably enough evidences for the confirmation of the applicant's identity, the term of one day will not begin (neither will the term of two days, established in art. 307 as solution to the lack of velocity of the certification authorities, because this second term has the same "dies a quo"). And then, the decision of effective revocation can be delayed excessively, and even indefinitely.

In conclusion, the different legal solutions try to solve this question demanding simply and generically diligence in the actions of the certification entity, once received the revocation request, with the problem of determining when there is and when there is not that diligence, question that probably should be solved according to the practice usages. Only one of this legal solutions (Utah Law) makes concrete a temporary period which is the key to move to the certification entity the liability; so that it is necessary to understand that, meanwhile, during this period, the holder is yet assuming the liability (and even, as we will see in next period, it is possible that the holder assumes the liability after the adoption of the decision of revocation, and even after including the revocation in the corresponding list of revoked certificates, while that list could not be known by the community of third users of certificates).

Art. 9.3 of Spanish Law, after linking in art. 9.2 the effectiveness of the extinction of the certificate to its publication, establishes, firstly, the certification authority's obligation of publishing the extinction of the effectiveness of the certificate in the register that, according to art. 11.e), obligatorily every certification entity must maintain (so does European Directive implicitly in art. 6.2). Contrary to other regulations, art. 9 of Spanish Law doesn't require velocity in the publication (neither does European Directive in art. 6); only art. 12 of Spanish Law and Annex II, b) of European Directive establish the requirement of assuring the extinction or suspension of the effectiveness of the certificates in a sure and immediate way, but this requirement is not applied to all kind of certification entities but only to those entities that issue a special kind of certificates (the so-called "qualified certificates"). Nevertheless, in spite of not demanding this velocity in the execution of the obligation of publishing the extinction of a certificate, art. 9.3 of Spanish Law establishes that the certification authority will be liable of the possible damages caused to the signatory, or to third parties in good faith, for the delay in the publication (for failure

to register revocation, according to art. 6. 2 of European Directive). Therefore, a velocity in the publication is demanded implicitly, because otherwise the certification entity will be liable in the event of delay. So, the liability of the certifier is linked to a concept with imprecise and uncertain content, with the difficulties and insecurities that it supposes.

Fourth Period. *The decision of revocation has been taken but it is possible that the user of the certificate cannot still know it.* In this period, the certification entity has already revoked the certificate and it has even included it in the corresponding list of revoked certificates or the directory of revoked certificates. Nevertheless, in function of the used communication system, it is possible that the community of users cannot still know the revocation. In this situation, the question of the distribution of the risk depends of the system of publication of revocation used (and probably agreed by the parties):

- in the case of the system of periodical lists of revoked certificates, the certificate user won't know the revocation until the corresponding periodical list of revocation will be issued. So that if, e.g., the decision of revocation is adopted when the number one list of revoked certificates (CRL1) has just been issued, the revocation won't be published until the number 2 list of revoked certificates (CRL2) will be issued. In these cases, considering the fact that the revocation has not been published and it cannot be known by third users, we understand that this revocation cannot be alleged to the third users of certificates: it has no effect in front of these third users. It would be in this way even in the extreme case that, for example, in a system of issuing periodical lists hourly, at 12:59 the user of a certificate trusts the list issued at 12:00, without waiting for the next list that must be issued at 13:00 o'clock, a new list that can announce that the certificate has been already revoked.

The problem is, again, who assumes the risk, and the derived responsibility of the use, as if it was valid, of a certificate that has been yet revoked, but whose revocation has not still been published. In order to give a solution to this problem we understand that the considerations and arguments exposed for the previous period can be used.

- in the case of the system of immediate publication of the revocation, it seems reasonable to think the user of the certificate knows the revocation during this fourth period, because there is not a period among the inclusion the revocation in the list and the publication of the list: in this system, the revocation can be known by the users in an instantaneous way. So, in this case, if the revocation has already been included in the directory of revoked certificates, this revocation produce effects in front of the third users, because these third parties have had the possibility of knowing it; therefore, the user of a certificate has the obligation or at least the burden of checking the status of the certificate; if he doesn't check it, he will have to assume this negative consequence: the revocation will produce effects on him.

Therefore, the doubts on distribution of risks and attribution of liabilities in this period would not exist if we will apply Spanish Law, and European Directive that, as we have exposed above, establish the requirement of assuring the extinction or suspension of the effectiveness of the certificates in a sure and immediate way; and, according to that requirement, it would not be acceptable non-immediate systems of communication of the revocation.

Nevertheless, this requirement is not applied to every certification authority but only to those entities that issue a special kind of certificates (the qualified certificates); for the other certification entities which don't issue such certificates, it is simply required a register of certificates with "the circumstances that affect to the suspension or loss of validity of their effects", without any requirement about the celerity and the immediate effects of the system. So the problem in this period remains at least for the certifiers issuing non-qualified certificates.

Last Period. *It is the period after the publication and effective diffusion of the revocation.* The certification entity has finished his function of distribution of information about the revocation: the corresponding periodical list has been issued, the list has been sent to the user or the revocation has been included in the corresponding directory. If a certificate user uses a revoked certificate in this last period, it can be reasonable attribute to the user the consequences of this use, because he has had opportunity of knowing the revocation since it has been object of effective diffusion and publication. Therefore, in definitive, the user of a certificate has the obligation or the burden of checking the validity of the certificate, otherwise, he will have to assume certain negative consequences: the derived damages of trusting a revoked certificate. This solution is clearly established in the different regulation on electronic signatures that, as we have seen, links the effects of the extinction to the publication.

Partial Conclusion. Summing up, the obligations and the liabilities of the certification entity, the holder and the certificate user in case of revocation of a certificate are a complex question that deserves a rigorous attention when establishing the practices of the certification entities and the agreements that will rule the relationships of the different parties. Attending the fact that in these first moments of development of the system of certificates, neither the practice nor the laws have still solved in a clear way the question of the rights and liabilities of every party (as we have seen legal rules are in general partial and insufficient), it is essential and indispensable that the definitive rules controlling the events of the revocation will be clearly established by the different parties.

Also, attending the fact that the liability of the different parties can vary in function of the temporary periods, the resolution of discussions and litigations on revocation will depend on the accuracy of the exact determination of the moment of the diverse facts and actions, so that the situation will improve if the signed transactions or the messages are digitally time-stamped. It is, in fact, the temporary problem of the system of certificates, that is specially important in case of revocation.

In this situation of certificate revocation it is important in the first place to prove the exact moment when the certificate was revoked in order to remove liability for contracts signed after this moment with the private key: if person A loses his private key and immediately cancels the certificate, he will not be liable, as we have explained, for the signatures the third illegitimate possessor of the key makes after that time of revocation, which will be deemed fraudulent. But it is equally important to be able to prove the time when the contracts were signed, because, if person B had robbed the private key of A and falsified signed messages with the same key and predated them to before the time of the robbery and the revocation, these messages would be attributed to A who could not allege that they had been signed after the time

of the robbery (after which the revocation of the corresponding certificate would have proceeded) and that he was not responsible for them. Also the owner of the private key, person A, could try and take advantage of the uncertainty over the time of the contract by alleging, after sending a signed message to B (now just a recipient of the message) that the message wasn't valid since it was created on a date after the loss of the private key. In other words the signatory of a particularly important message could fraudulently deny his/her own signature, declaring simply the compromise of the private key, and alleging that the key had been yet compromised when the signature was generated. Summing up, the problem of the time in the certificate system is especially grave in cases of revocation and, however, it is not solved in the different legislations: in the case of European Directive and Spanish Law, it's true that both laws establish that certification service providers issuing qualified certificates must ensure that the date and the time when a certificate is issued or revoked can be determined precisely, but they don't require the precise determination of the time of signing an electronic message. So, the temporary problem remains unsolved.

3 Limits and Extension of Liability

Once established, in general, the liability of the certification service provider for the accuracy of the content of the certificate (European Directive, art. 6.1.a; Spanish Law, art. 14.1); and, in particular, for failure to register revocation (European Directive, art. 6.2) or delay in publication (Spanish Law, art. 9.3), it must not be forgotten that the liability of the certifier without a specific legislation is contained under the general regulations for liability, contractual or extra-contractual, and is, as has been shown, potentially high and unforeseeable, given the indefinite number of operations covered by a same certificate.

Because of this, and in order to stimulate the development of certification authorities which could be halted in cases of unforeseeable or unlimited risk, different legislations and legislative initiatives do expressly admit or favorably contemplate the existence of possible limitations to the liability of the providers of certification services (it must be pointed out that the existence of limitations to liability of the subscribers and the users of the certificates can be equally as necessary). In the same fashion the European Directive on electronic signatures, art. 6, and the Spanish Law, art. 14, establish distinct limits to liability which benefit the providers of certification services, and which are, basically, as follows.

3.1 Qualitative Limitations

There are two main kinds of qualitative limitations: the subjective nature of liability and the possible limitations of use of a certificate.

Subjective and Strict Liability. Art. 6.1 of Directive establishes that the providers of certification services will be held liable for the accuracy of all information in the qualified certificate, without stating, for now, the nature of that responsibility.

However, in spite of what was established in art. 6.1, art. 6.2 of the proposal of Directive initially presented by the Commission said that "Member States shall ensure

that a certification service provider is not liable for errors in the information in the qualified certificate that has been provided by the person to whom the certificate is issued, if it can demonstrate that it has taken all reasonably practicable measures to verify that information“.

And, finally, in the definitive version of the Directive approved, the content of this art. 6.2 has been eliminated, and has been added a new ending to article 6.1 that is as follows: “unless the certification service provider proves that he has not acted negligently”.

In the same way, art. 14.1 of Spanish law establishes the general rule of liability stating that lenders of certifications services will be held liable for damages caused to any person in the exercise of the activity of certification, but only, according to the ending of this art. 14.1, “when they act negligently”; and the certification authority must prove that he acts diligently.

So, in the debated alternative between strict or subjective liability of the certification entity caused by false or erroneous certificates, the European Directive and the Spanish Law (and also other legislations on electronic signatures) adopt the solution that the liability is held by that entity only when it has been negligent or non-diligent (meaning that if the certifier had not been negligent, the user who had trusted in the erroneous certificate could not act against that certifier but only against the subscriber or end up assuming the consequences of the error himself). Nevertheless, and taking into account how difficult it would be for a user to prove negligence on the part of the certifying entity, the burden of proof rests on the certifying entity if it wants to exonerate itself of the liability. With article 6.2 of the first proposal of Directive presented by the Commission, the certification service provider had to prove itself diligent if it wants to exonerate itself of the liability; and this is also the content of art. 14.1 of Spanish Law; that is, if unable to prove their diligence, the providers of certification services must be held liable for erroneous certificates. But in the final version of Directive, as we have seen, the content of this article 6.2 has been suppressed and has appeared a new content at the ending of art. 6.1; and according to this new ending, if the certification service provider wants to exonerate himself of liability, he had to prove itself not diligent but only non-negligent; that is, only if unable to prove their non-negligence, the providers of certification services must be held liable for erroneous certificates.

In the same way, in case of revocation of a certificate, the European Directive establishes that the providers of certification services will be held liable for failure to register revocation (art. 6.2), without stating, for now, the nature of that responsibility. But the ending of article 6.2 is the same as art. 6.1: “unless the certification service provider proves that he has not acted negligently”. So, only if unable to prove their non-negligence, the providers of certification services must be held liable for failure to register revocation.

Limitations to possible use. Article 6.3 of the European Directive establishes that the Member States shall ensure that certification service provider may indicate in a qualified certificate limitations on the uses of a certain certificate. Therefore, the certifier may also limit the liability associating the certificate with certain uses or in certain areas of activity, such as is indicated in art. 6.3 which states that the authority is not liable for uses of the certificate which fall outside those specified or established uses (for example, the use of the complementary private and public keys certified for

specified operations, or the use of the certificate in the field of the organization for which it has been generated).

In the same way, as most legislations on electronic signatures, art. 14.2 of Spanish Law states that a certification entity will be held liable of damage caused for non-authorized uses of a qualified certificate, only when the limit of use has not been included clearly in the content of the certificate in order that the third users of the certificate will be able to know the existence of this limit. So, it is necessary to include the limit in the content of the certificate; if it is not included, the certification authority will not be exonerate of liability for non-authorized uses. In the same sense, art. 6.3 of European Directive requires that these limits must be “recognizable to third parties”.

3.2 Quantitative Limitations

Together with qualitative limitations, most legislations on electronic signatures admit quantitative limitations of liability for certification authorities. The European Directive rules a possible quantitative limitation to liability in the following terms (art. 6.4): “The Member States shall ensure that a certification service provider may indicate in the qualified certificate a limit on the value of transactions for which the certificate can be used, provided that the limit is recognizable to third parties”. And also art. 14.2 of Spanish Law states the possibility of including in a qualified certificate that kind of quantitative limits referred to the value of transactions.

We consider these articles confusing, because the value of the transactions and the value of the damages that may be suffered by the user of the certificate are distinct and non-equivalent concepts. Furthermore, they limit, but not totally, the risk and potential liability of an authority of certification in the sense of establishing the quantity of each transaction covered by the use of the certificate (for example, transactions to the value of less than 1000 Euros) which effectively limits the risk of the transaction but not the total potential risk derived from the certificate, which could continue to be used on an unlimited number of occasions (the solution for this lies in the transactional certificates valid and specific for one or more concrete transactions, and the digital signatures of those limited transactions).

This quantitative limit, with its confused wording, can be also understood to mean that the total value of the transactions are limited (for example, transactions which, when totaled, do not surpass 1 million Euros), an obviously favorable condition for the provider of the certification service, but detrimental to the users, who would still find it difficult to know, in the moment of verification of a signature with a certificate with a limit of this kind, the total value of the accumulated transactions whose digital signatures have been verified with this same certificate, in order to judge whether or not the fixed limit had been exceeded, and determine if they could claim against the liability of the service provider. And this poses also the question of the priority of the claimants.

Summing up, the legal establishment of limits of liability could effectively aid the development of certification activities. In practical terms, a limit of liability could combine with a class of certificate which in turn would entail a level of surety and corresponding cost structure. With this model, the different limits are a function of the relative security of each class of certificate.

Nevertheless, in the case of the European Directive and the Spanish Law that we are analyzing, it can be seen that all the limitations to liability clearly favor the providers of certification services, almost to the point of excess, especially when one takes into account the transference of the assumption of risk to other implicated parties (if the damages exceed the maximum limit, the relying party, the third user of the certificate, must take up the excess), and to whom, in exchange, are given no fixed limits or provisions. This appears unjust, especially when this other party is the consumer. At this point it is fitting to suggest how important the establishment of a common fund would be to cover just this type of situation, a fund created from contributions from all the certification authorities and which would be made more viable by the application of a licensing system for providers of certification services.

On the other hand, in order to inspire confidence a certificate would have to establish or incorporate, amongst other things, the degree of investigation developed by the authority to confirm the declarations made by the signatory, and also the limit recommended for the transactions based on the certificate (in this sense, the Directive, art. 6.3 and 6.4, and also the Spanish law, art. 14, state that the quantitative limit must be recognizable to third parties, requirement non established in the first proposal of Directive presented by the Commission). It also may be useful to homologate the certificates; that is to say, establish classes or categories of certificates generally accepted by the certifying entities and widely known throughout the community of users, in which would be fixed definite types of certificate issued by each authority, permitting the user an approximate idea of the value of each certificate.

4 Conclusions

In conclusion, the following observations may be made about the liability of certification authorities from a legal point of view, according to the rules on liability of the European Directive, and the Spanish Law, which in general we consider incomplete and excessive, without taking into proper account the necessary balance of all the parties of the system of certificates:

1. In the first place, we consider these rules partial and incomplete because they rule only the liability of the certification authority, when, as has been said, it could also create a strictly objective liability for the subscriber (for example, with the loss of the private key) and, second, because they solely rule liability with regard to the exact contents of the certificate (ignoring, as has been explained, other situations which can also generate liability; it's true that there are rules of liability in case of revocation, but, as we have explained before, the regulation is not complete). Similarly, the different relationships that exist between the various persons of the certification system are not completely regulated. There is no definition of the rights and obligations of the parties except in the approach to the subject of the liability of the provider of the certification service with respect to the third user of the certificate.
2. Regarding a question as important as the revocation of a certificate, the Directive and also the Spanish Law, are limited to demanding a rapid and secure system of revocation without properly considering the distribution and assignation of liability among the different parties (certifying entity, subscriber and third user) during the period of the revocation (from, for example, the date of compromise of the private

key to the effective publication of the revocation of that key); in that sense, the content of these rules is partial and incomplete. Neither is there offered any options between the different technically possible ways to publicize the revocations (certificate revocation lists, broadcast revocation lists or immediate revocation) which can suppose different distributions of liability.

3. The obligations and the liabilities of the certification entity, the holder and the certificate user in case of revocation of a certificate are a complex question that deserves a rigorous attention when establishing the practices of the certification entities and the agreements that will rule the relationships of the different parties. Attending the fact that in these first moments of development of the system of certificates, neither the practice nor the laws have still solved in a clear way the question of the rights and liabilities of every party (as we have seen legal rules are in general partial and insufficient), it is essential and indispensable that the definitive rules controlling the events of the revocation will be clearly established by the different parties.
4. Attending the fact that the liability of the different parts can vary in function of the temporary periods, the resolution of discussions and litigations on revocation will depend on the accuracy of the exact determination of the moment of the diverse facts and actions, so that the situation will improve if the signed transactions or the messages are digitally time-stamped. It is, in fact, the temporary problem of the system of certificates, that is specially important in case of revocation. However, this problem is not solved in the different legislations: in the case of European Directive and Spanish Law, it's true that both laws establish that certification service providers issuing qualified certificates must ensure that the date and the time when a certificate is issued or revoked can be determined precisely, but they don't require the precise determination of the time of signing of an electronic message. So, the temporary problem remains unsolved.
5. Finally, we consider legal rules excessive because of the qualitative and quantitative limitations established for the liability of certification service providers: subjective liability of the certifier for direct damages and with possible limits of quantity; meanwhile the subscriber according to the general rules of law (and there being no regulations made for the subscriber in European Directive and Spanish Law) may have to unreasonably assume an unlimited and objective liability. It's true the legal establishment of limits of liability could effectively aid the development of certification activities. Nevertheless, in the case of the European Directive and the Spanish Law it can be seen that all the limitations to liability clearly favor the providers of certification services, almost to the point of excess, especially when one takes into account the transference of the assumption of risk to other implicated parties (if the damages exceed the maximum limit, the relying party must take up the excess), and to whom, in exchange, are given no fixed limits or provisions. This appears unjust, especially when this other party is the consumer. At this point it is fitting to suggest how important the establishment of a common fund would be to cover just this type of situation, a fund created from contributions from all the certification authorities and which would be made more feasible by the application of a licensing system for providers of certification services.

References

1. ABA (American Bar Association), *Digital signature guidelines, Legal infrastructure for Certification Authorities and secure electronic commerce*, Information Security Committee, Electronic Commerce and Information Technology Division, Section of Science and Technology, August 1, 1996, USA.
2. Commission of the European Communities, Proposal of Directive of the European Parliament, and the Council for a common framework on electronic signatures (COM (1998) 297 final).
3. Commission of the European Communities, Amended proposal for a European Parliament and Council Directive on a common framework for electronic signatures, (COM(1999) 195 final).
4. European Parliament and European Council, Directive 1999/99/CE of the European Parliament and Council Directive on a common framework for electronic signatures, (13-12-1999).
5. Ford W.- Baum M. S., *Secure electronic commerce*, 1997 (second edition, 2001).
6. Real Deceto-Ley 14/1999, de 14 de septiembre, de firma electrónica, (Spanish Electronic Signature Law), 1999.
7. Regolamento contenente modalità di applicazione dell'articolo 15, comma 2, della legge 15 marzo 1997, n. 59, in formazione, archiviazione e trasmissione di documenti con strumenti informatici e telematici (Italian rules on digital signatures), 1997.
8. UNCITRAL (Commission of the United Nations for the International Commercial Law), *Model Law on Electronic commerce*, 1997.
9. *Utah Digital Signature Law*, 1995 (later modified in 1997 and 2000).

Experimental Testing of the Gigabit IPSec-Compliant Implementations of Rijndael and Triple DES Using SLAAC-1V FPGA Accelerator Board

Pawel Chodowiec¹, Kris Gaj¹, Peter Bellows², and Brian Schott²

¹ Electrical and Computer Engineering, George Mason University, 4400 University Drive,
Fairfax, VA 22030

{kgaj, pchodow1}@gmu.edu

² University of Southern California - Information Sciences Institute
Arlington, VA 22203

{pbellows, bschott}@east.isi.edu

Abstract. In this paper, we present the results of the first phase of a project aimed at implementing a full suite of IPSec cryptographic transformations in reconfigurable hardware. Full implementations of the new Advanced Encryption Standard, Rijndael, and the older American federal standard, Triple DES, were developed and experimentally tested using the SLAAC-1V FPGA accelerator board, based on Xilinx Virtex 1000 devices. The experimental clock frequencies were equal to 91 MHz for Triple DES, and 52 MHz for Rijndael. This translates to the throughputs of 116 Mbit/s for Triple DES, and 577, 488, and 423 Mbit/s for Rijndael with 128-, 192-, and 256-bit keys respectively. We also demonstrate a capability to enhance our circuit to handle the encryption and decryption throughputs of over 1 Gbit/s regardless of the chosen algorithm. Our estimates show that this gigabit-rate, double-algorithm, encryption/decryption circuit will fit in one Virtex 1000 FPGA taking approximately 80% of the area.

1 Introduction

IPSec is a set of protocols for protecting communication through the Internet at the IP (Internet Protocol) Layer [15, 22]. One of the primary applications of this protocol is an implementation of Virtual Private Networks (VPNs). In IPSec Tunnel Mode, multiple private local area networks are connected through the Internet as shown in Fig. 1a. Since the Internet is an untrustworthy network, a secure tunnel must be created between security gateways (such as firewalls or routers) belonging to private networks involved in the communication. The information passing through the secure tunnel is encrypted and authenticated. Additionally, the original IP header, containing the sender's and receiver's addresses is also encrypted, and replaced by a new header including only information about the security gateway addresses. This way a limited resistance against the traffic control analysis is accomplished. A second use of IPSec is client-to-server or peer-to-peer encryption and authentication (see Fig. 1b). In IPSec Transport Mode, many independent pair-wise encryption sessions may exist

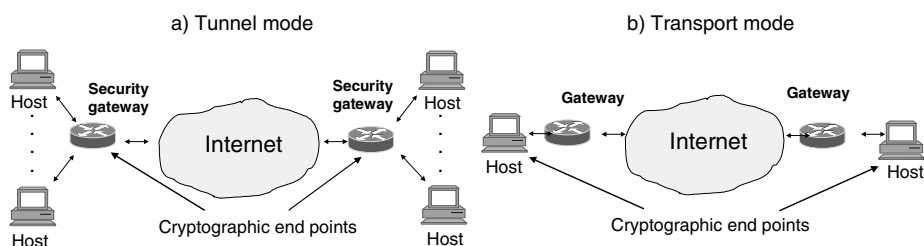


Fig. 1. IPSec Tunnel and Transport Modes

simultaneously. *The large number of connections and high bandwidth supported by a single security gateway or server suggests the use of hardware accelerators for implementing cryptographic transformations.*

The suite of cryptographic algorithms used for encryption and authentication in IPSec is constantly evolving. In the case of encryption, current implementations of IPSec are required to support DES, and have the option of supporting Triple DES, RC5, IDEA, Blowfish, and CAST-128. Since DES has been shown to be vulnerable to an exhaustive key-search attack using the computational resources of a single corporation [2], the current implementations of IPSec typically support Triple DES. In 1997, the National Institute of Standards and Technology (NIST) initiated an effort towards developing a new encryption standard, called AES (Advanced Encryption Standard) [1]. The development of the new standard was organized in the form of a contest coordinated by NIST. In October 2000, Rijndael was announced as the winner of the contest and a future Advanced Encryption Standard. In November 2000, a first Internet-draft was issued, proposing including AES-Rijndael as a required encryption algorithm in IPSec, with the remaining AES contest finalists as optional algorithms to be used in selected applications [11].

An encryption algorithm is not the only part of IPSec that is currently being extended and modified. Other modifications currently being considered include different modes of operation for encryption algorithms [18], hash functions used by authentication algorithms [21], type and parameters of public key cryptosystems used by a key management protocol, etc. *The fast and hard to predict evolution of IPSec algorithms leads naturally to prototype and commercial implementations based on reconfigurable hardware.*

An FPGA implementation can be easily upgraded to incorporate any protocol changes without the need for expensive and time-consuming physical design, fabrication, and testing required in case of ASICs. Additional capabilities appear when an FPGA accelerator supports a real-time partial reconfiguration. In this case, the accelerator can reconfigure itself on the fly to adapt to

- traffic conditions (e.g., by changing the number of packet streams processed simultaneously),
- phase of the protocol (e.g., by using the same FPGA with time sharing for implementing key exchange, encryption, and authentication),
- various key sizes and parameter values (e.g., by adjusting the circuit architecture to different key sizes and different values of system parameters).

Additionally, several optional IPSec-compliant encryption, authentication, and key exchange algorithms can be implemented, and their bitstreams stored in the cache

memory on the FPGA board. Algorithm agility accomplished this way can substantially increase the system interoperability.

In this paper, we present the results of the first phase of our project aimed at implementing a full suite of IPSec cryptographic transformations using SLAAC-1V FPGA board. In this phase, two encryption algorithms AES-Rijndael and Triple DES were implemented and experimentally tested in our environment.

2 FPGA Board

The SLAAC-1V PCI board is an FPGA-based computation accelerator developed under a DARPA-funded project called Systems-Level Applications of Adaptive Computing (SLAAC). This project, led by USC Information Sciences Institute (ISI), investigated the use of adaptive computing platforms for open, scalable, heterogeneous cluster-based computing on high-speed networks. Under the SLAAC project, ISI developed several FPGA-based computing platforms and a high-level distributed programming model for FPGA-accelerated cluster computing [16]. About a dozen universities and research labs are using SLAAC-1V for a variety of signal and image processing applications.

The SLAAC-1V board architecture is based on three user-programmable Xilinx Virtex XCV-1000-6 FPGA devices. Each of these devices is composed of 12,288 basic logic cells referred to as CLB (Configurable Logic Block) slices, and includes 32 4-kbit blocks of synchronous, dual-ported RAM. All devices can achieve synchronous system clock rates up to 200 MHz, including input/output interface.

The logical architecture of SLAAC-1V is shown in Fig. 2. The three Virtex 1000 FPGAs (denoted as X0, X1, and X2) are the primary processing elements. They are connected by a 72-bit “ring” path as well as a 72-bit shared bus. The width of both buses supports an 8-bit control tag associated with each 64-bit data word. The direction of each line of both buses can be controlled independently. The processing elements are connected to ten 256K x 36-bit SRAMs (Static Random Access Memories) located on mezzanine cards. The FPGAs X1 and X2 are each connected to four SRAMs, while X0 is connected to two. The memory cards have passive bus switches that allow the host to directly access all memories through X0.

About 20% of the resources in the X0 FPGA are devoted to the PCI interface and board control module. The remaining logic of this device (as well as the entire X1 and X2 FPGAs) can be used by the application developer. The 32/33 control module release uses the Xilinx 32-bit 33MHz PCI core. The control module provides high-speed DMA (Direct Memory Access), data buffering, clock control (including single-stepping and frequency synthesis from 1 to 200 MHz), user-programmable interrupts, etc. The current 32/33 control module has obtained DMA transfer rates of over 1 Gbit/s (125 MB/s) from the host memory, very near the PCI theoretical maximum. The bandwidth for SLAAC-1V using the 64-bit 66MHz PCI controller (using the Xilinx 64-bit 66MHz core) has been measured at 2.2 Gbit/s. The user’s design located in X0 is connected to the PCI core via two 256-deep, 64-bit wide FIFOs. The DMA

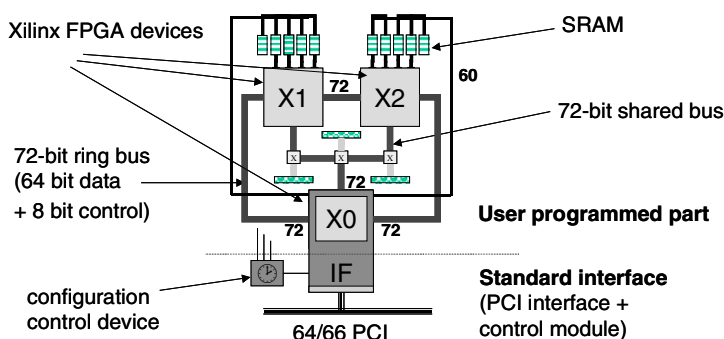


Fig. 2. SLAAC-1V Architecture

controller located in the interface part of X0 can transfer data to or from these FIFOs as well as to provide fast communication between the host and the board SRAMs. The DMA controller load balances input and output FIFOs and can process large memory buffers without host processor interaction. Current interface development includes managing memory buffer rings on the FPGA to minimize host interrupts on small buffers.

SLAAC-1V supports partial reconfiguration, in which part of an FPGA is reconfigured while the rest of the FPGA remains active and continues to compute. A small dedicated Virtex 100 configuration control device is used to configure all FPGAs and manages 6 MB of flash / SRAM as a configuration “cache”.

The work discussed in this paper was done in collaboration with the ISI Gigabit-Rate IPSec (GRIP) project, which is funded in the DARPA Next Generation Internet (NGI) program. The GRIP team has constructed a gigabit Ethernet daughter card which connects to SLAAC-1V in place of the crossbar connection of the X0 chip. To the host, the SLAAC-1V / GRIP system appears to be a gigabit Ethernet card with optional acceleration features. The GRIP team is currently customizing the TCP/IP stack for the Linux operating system to take advantage of the hardware acceleration in order to deliver fully-secure, fully-authenticated gigabit-rate traffic to the desktop.

3 Implementation of Rijndael

Rijndael is a symmetric key block cipher with a variable key size and a variable input/output block size. Our implementation supports all three key sizes required by the draft version of the AES standard, 128, 192, and 256 bits. Our key scheduling unit is referred to as 3-in-1, which means that it can process all three key sizes. Switching from one key size to the other is instantaneous, and is triggered by the appropriate control signals. Our implementation is limited to the block size of 128-bits, which is the *only* block size required by Advanced Encryption Standard. Implementing other block sizes, specified in the original, non-standardized description of Rijndael is not justified from the economical point of view, as it would substantially increase circuit area and cost without any substantial gain in the cipher security.

Rijndael is a substitution-linear transformation cipher based on S-boxes and operations in the Galois Fields. Below we describe the way of implementing all component operations of Rijndael, and then present how these basic operations are combined together to form the entire encryption/decryption unit.

3.1 Component Operations

Implementation of the encryption round of Rijndael requires realization of four component operations: ByteSub, ShiftRow, MixColumn, and AddRoundKey. Implementation of the decryption round of Rijndael requires four inverse operations InvByteSub, InvShiftRow, InvMixColumn, and AddRoundKey.

ByteSub is composed of sixteen identical 8x8 S-boxes working in parallel. **InvByteSub** is composed of the same number of 8x8-bit inverse S-boxes. Each of these S-boxes can be implemented independently using a 256 x 8-bit look-up table.

A Virtex XCV-1000 device contains 32 4-kbit Block Select RAMs. Each of these memory blocks is a synchronous, dual-ported RAM with the data port width configurable to an arbitrary power of two in the range from 1 to 16. Each memory block can be used to realize two table look-ups per clock cycle, one for each data port.

In particular, each 4-kbit Block Select RAM can be configured as a 512 x 8-bit dual-port memory. If encryption or decryption are implemented separately, only the first 256 bytes of each memory block are utilized as a look-up table. If encryption and decryption are implemented together within the same FPGA, both uninverted and inverted 256 byte look-up tables are placed within one memory block. In each case, 16 data bits are processed by one memory block, which means that a total of 8 memory blocks are needed to process the entire 128-bit input.

ShiftRow and **InvShiftRow** change the order of bytes within a 16-byte (128-bit) word. Both transformations involve only changing the order of signals, and therefore they can be implemented using routing only, and do not require any logic resources, such as CLBs or dedicated RAM.

The **MixColumn** transformation can be expressed as a matrix multiplication in the Galois Field $GF(2^8)$:

$$\begin{bmatrix} B_0 \\ B_1 \\ B_2 \\ B_3 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} A_0 \\ A_1 \\ A_2 \\ A_3 \end{bmatrix} \quad (1)$$

Each symbol in this equation (such as A_i , B_i , '03') represents an 8-bit element of the Galois Field. Each of these elements can be treated as a polynomial of degree seven or less, with coefficients in $\{0,1\}$ determined by the respective bits of the $GF(2^8)$ element. For example, '03' is equivalent to '0000 0011' in binary, and to

$$C(x) = 0 \cdot x^7 + 0 \cdot x^6 + 0 \cdot x^5 + 0 \cdot x^4 + 0 \cdot x^3 + 0 \cdot x^2 + 1 \cdot x + 1 \cdot 1 = x + 1 \quad (2)$$

in the polynomial basis representation.

The multiplication of elements of $GF(2^8)$ is accomplished by multiplying the corresponding polynomials modulo a fixed irreducible polynomial

$$m(x) = x^8 + x^4 + x^3 + x + 1. \quad (3)$$

For example, multiplying a variable element $A = a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0$ by a constant element '03' is equivalent to computing

$$\begin{aligned} B(x) &= b_7 x^7 + b_6 x^6 + b_5 x^5 + b_4 x^4 + b_3 x^3 + b_2 x^2 + b_1 x + b_0 = (4) \\ &= (a_7 x^7 + a_6 x^6 + a_5 x^5 + a_4 x^4 + a_3 x^3 + a_2 x^2 + a_1 x + a_0) \cdot (x+1) \\ &\quad \text{mod } (x^8 + x^4 + x^3 + x + 1). \end{aligned}$$

After several simple transformations

$$\begin{aligned} B(x) &= (a_7 + a_6) x^7 + (a_6 + a_5) x^6 + (a_5 + a_4) x^5 + (a_4 + a_3 + a_7) x^4 + (a_3 + a_2 + a_7) x^3 + \\ &\quad + (a_2 + a_1) x^2 + (a_1 + a_0 + a_7) x + (a_0 + a_7), \end{aligned} \quad (5)$$

where '+' represents an addition modulo 2, *i.e.* an XOR operation.

Each bit of a product B, can be represented as an XOR function of at most three variable input bits, *e.g.*, $b_7 = (a_7 + a_6)$, $b_4 = (a_4 + a_3 + a_7)$, etc.

Each byte of the result of a matrix multiplication (1) is an XOR of four bytes representing the Galois Field product of a byte A_0 , A_1 , A_2 , or A_3 by a respective constant. As a result, the entire MixColumn transformation can be performed using two layers of XOR gates, with up to 3-input gates in the first layer, and 4-input gates in the second layer. In Virtex FPGAs, each of these XOR operations requires only one lookup table (*i.e.*, a half of a CLB slice).

The **InvMixColumn** transformation can be expressed as a following matrix multiplication in $GF(2^8)$.

$$\begin{bmatrix} A_0 \\ A_1 \\ A_2 \\ A_3 \end{bmatrix} = \begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} B_0 \\ B_1 \\ B_2 \\ B_3 \end{bmatrix} \quad (6)$$

The primary differences, compared to MixColumn, are the larger hexadecimal values of the matrix coefficients. Multiplication by these constant elements of the Galois Field leads to the more complex dependence between the bits of a variable input and the bits of a respective product. For example, the multiplication $A = '0E' \cdot B$, leads to the following dependence between the bits of A and B:

$$a_7 = b_7 + b_6 + b_5 + b_4 \quad (7)$$

$$a_6 = b_6 + b_5 + b_4 + b_3 + b_7 \quad (8)$$

$$a_5 = b_5 + b_4 + b_3 + b_2 + b_6 \quad (9)$$

$$a_4 = b_4 + b_3 + b_2 + b_1 + b_5 \quad (10)$$

$$a_3 = b_3 + b_2 + b_1 + b_0 + b_6 + b_5 \quad (11)$$

$$a_2 = b_2 + b_1 + b_0 + b_6 \quad (12)$$

$$a_1 = b_1 + b_0 + b_5 \quad (13)$$

$$a_0 = b_0 + b_7 + b_6 + b_5. \quad (14)$$

The entire InvMixColumn transformation can be performed using two layers of XOR gates, with up to 6-input gates in the first layer, and 4-input gates in the second layer. Because of the use of gates with the larger number of inputs, the InvMixColumn transformation has a longer critical path compared to the MixColumn transformation, and the entire decryption is more time consuming than encryption.

AddRoundKey is a bitwise XOR of two 128-bit words and can be implemented using one layer of 128 look-up tables, which translates to 64 CLB slices. Assuming that one operand of the bitwise XOR is fixed, this operation is an inverse of itself, so no special transformation is required for decryption.

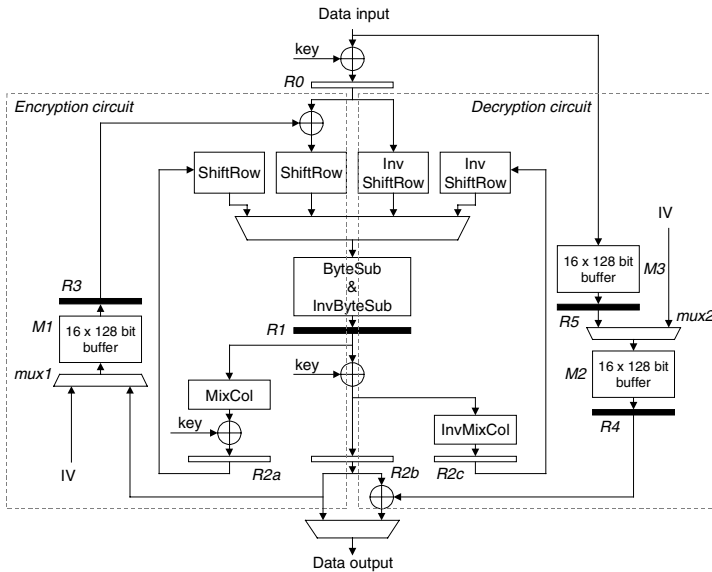


Fig. 3. The general architecture of Rijndael

3.2 General Architecture of the Encryption/Decryption Unit

The block diagrams of the encryption/decryption unit in the basic iterative architecture and in the extended pipelined architecture are shown in Fig. 3. Only registers R1, R3, R4, and R5 (shaded rectangles in Fig. 3) are present in the basic iterative architecture. The remaining registers (transparent rectangles in Fig. 3) have been added in the extended architecture based on the concept of inner-round pipelining.

The register R1 is a part of Block SelectRAM, the synchronous dedicated memory, used to implement ByteSub and InvByteSub transformations, so it was chosen as a basic register in the basic iterative architecture. In this architecture, 11, 13, and 15 clock cycles are required in order to process one block of data for 128-, 192-, and 256-bit keys respectively. The critical path is located in the *decryption circuit*, and includes AddRoundKey (an xor operation), InvMixColumn, InvShiftRow, multiplexer, and InvByteSub (memory read). It is important to note that our *decryption circuit* has a structure (order of operations) similar to the *encryption circuit*, but still does not require any additional processing of round keys (unlike the architecture suggested in [5] and adopted in [8, 9, 10]).

Introducing pipeline registers R2a-c and R0 allows the circuit to process two independent streams of data at the same time. Our architecture assumes the use of the Cipher Block Chaining (CBC) mode for processing long streams of data. The CBC mode is the only mode required by the current specification of IPSec to be used with DES and all optional IPSec encryption algorithms. It is also most likely to be the first mode recommended for use together with AES. The encryption and decryption

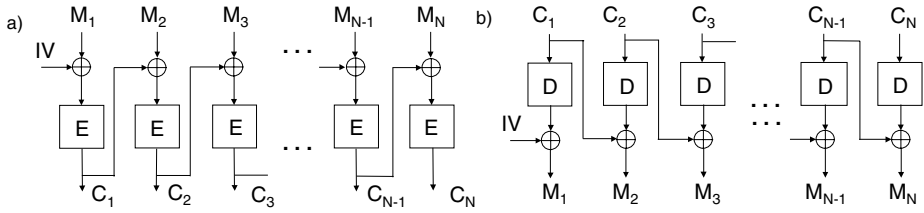


Fig. 4. Cipher Block Chaining Mode a) encryption, b) decryption

in the CBC mode are shown in Fig. 4. An initialization vector IV is different for each packet and is transmitted in clear as a part of the packet header. The CBC mode allows concurrent encryption of blocks belonging to different packets, but not to the same packet. This limitation comes from the fact that the encryption of any block of data cannot begin before the ciphertext of the previous block becomes available (see Fig. 4a). The same limitation does not apply to decryption, where all blocks can be processed in parallel.

In our implementation, the memory buffers M_1 , M_2 , and M_3 are used to store the last (*i.e.*, the most recently processed) ciphertext blocks for up to 16 independent streams of data. Before the processing of the given stream begins, the corresponding memory location is set to the initialization vector used during the encryption or decryption of the first block of data.

Our architecture allows the simultaneous encryption of two blocks belonging to two different packets, and the simultaneous decryption of two blocks belonging to the same packet or two different packets.

The new secret-key block cipher modes, currently under investigation by NIST, are likely to allow unlimited parallel encryption and decryption of blocks belonging to the same packet [18]. An example of such a mode, likely to be adopted by NIST in the near future, is a counter mode [17]. Our implementation will be extended to permit such new modes as soon as they become adapted as draft standards.

Our architecture can be extended by adding additional outer-round pipeline stages, or implementing multiple instantiations of the same encryption/decryption unit, and using them for parallel processing of data. The total throughput in these extended architectures is directly proportional to the amount of resources (CLB slices, dedicated RAMs) devoted to the cryptographic transformations.

3.3 Round Key Module

The round key module consists of the 3-in-1 key scheduling unit and 16 banks of round keys. The banks of round keys are implemented using 8 Block SelectRAMs configured as two memories 256 x 64 bits. These memories permit storing up to 16 different sets of round keys, with 16 consecutive memory locations reserved for each set. Each set of subkeys may correspond to a different main key and a different security association.

The 3-in-1 key scheduling unit of Rijndael is shown in Fig. 5a. The operation of the circuit is described by formulas given in Fig. 5b. The unit is capable of computing two 32-bit words of the key material (w_i and w_{i+1}) per one clock cycle, independently

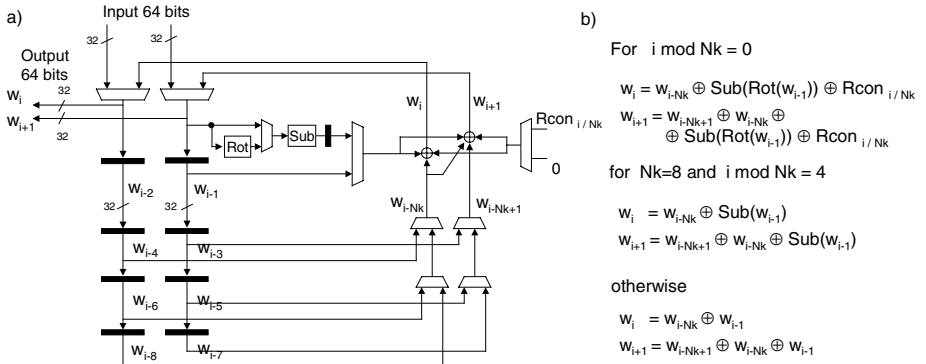


Fig. 5. The 3-in-1 key scheduling unit of Rijndael: a) the main circuit, b) formulas describing the operation of the circuit

of the size of the main key. Since each round key is 128 bit long (the size of the input block), two clock cycles are required to calculate each round key. Therefore, our key scheduling unit is not designed for computing subkeys on the fly. Instead, all round keys corresponding to the new main key are computed in advance and stored in one of the memory banks. This computation can be performed in parallel with encrypting data using previous main key, therefore key scheduling does not impose any performance penalty.

4 Implementation of Triple DES

4.1 Basic Architecture

In order to realize the Triple DES encryption and decryption it is sufficient to implement only one round of DES, as shown in Fig. 6a. The multiplexers *mux1* and *mux2* permit loading new data block or feed back the result of the previous iteration. Only half of the data block is transformed in each iteration, and this transformation depends on a round key coming from the key module. The DES-specific transformation function *F* has been implemented as a combinational logic and directly follows the algorithm specification. The multiplexers *mux3* and *mux4* choose the right feedback for consecutive iterations. In the single DES implementation, these multiplexers would not be required, because the feedback is always the same. However, this is not the case for Triple DES because of the data swapping at the end of the last round of DES. This feature becomes important when switching between the first and the second, and between the second and the third DES encryption in Triple DES. Performing the Triple DES encryption or decryption of one data block in the CBC mode requires 48 clock cycles, exactly as many as the number of the cipher rounds.

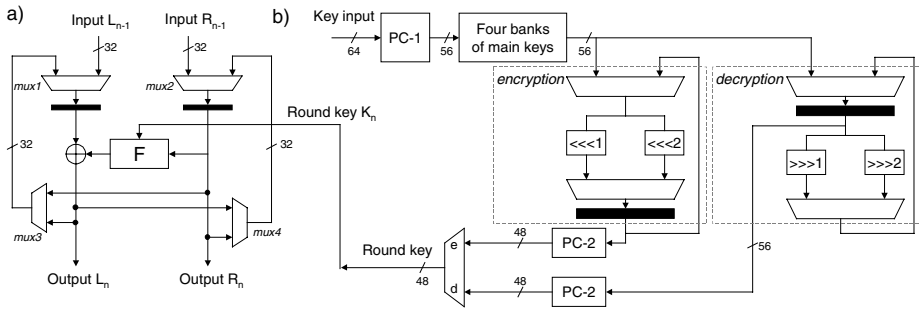


Fig. 6. Basic iterative architecture of Triple DES a) encryption/decryption unit, b) key scheduling unit

4.2 Round Key Module

The DES key schedule, which serves as a basis for the Triple DES key schedule, consists of very simple operations. Consecutive round keys are computed by rotating two halves of the main 56-bit key by one or two positions depending on the number of the round. The result of each next rotation goes through the Permuted Choice-2 function (PC-2), which selects 48 bits of a round key. Since DES key scheduling requires much simpler operations than encryption/decryption unit, it can be easily performed on the fly. This way only three 56-bit keys need to be stored on-chip. Our Triple DES key scheduling unit is shown in Fig. 6b.

Four banks of the key memories are placed at the input to the key scheduling circuit. Each bank contains three DES keys used by Triple DES. The user supplies 64-bit keys to the circuit, but only 56-bits of each key are selected by the Permuted Choice-1 function (PC-1) and stored in one of the memory banks. Each memory bank can hold all three keys required for performing Triple DES. All memory banks are built using dual-port memory, and can operate independently. They are organized in a way that permits writing new key to one of the banks, while any other bank may be used for the round key computations. The output of the round key memory goes to two simple circuits, one computes keys for encryption, the other for decryption.

4.3 Extended Architecture

We are currently in the process of developing an extended pipelined architecture of Triple DES. Our goal is to obtain throughput over 1 Gbit/s. Our approach is to fully unroll single DES and introduce pipeline registers between cipher rounds, as shown in Fig. 7. This leads to a capability of processing up to 16 independent data streams, which gives a throughput of around 1.5 Gbit/s. We should be able to maintain clock frequency at the similar or even greater level, since this architecture permits significant simplifications compared to the basic iterative architecture. Namely, multiplexers *mux3* and *mux4* are no longer required in any of the stages (see Fig. 6b), and key scheduling can be greatly simplified as shown in Fig. 7b.

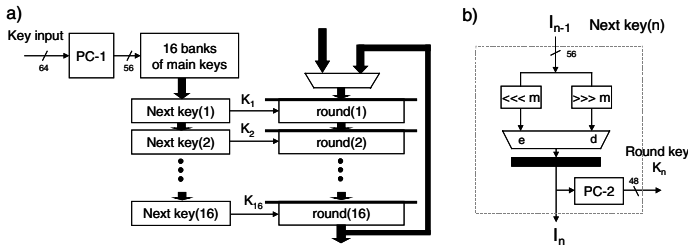


Fig. 7. Extended architecture of Triple DES: a) main circuit, b) next key module; the number of rotations m depends on the round number n , and can be equal to 0, 1, or 2

5 Testing Procedure

Our testing procedure is composed of three groups of tests. The first group is aimed at verifying the circuit functionality at a single clock frequency. The goal of the second group is to determine the maximum clock frequency at which the circuit operates correctly. Finally, the purpose of the third group is to determine the limit on the maximum encryption and decryption throughput, taking into account the limitations of the PCI interface.

Our first group of tests is based on the NIST Special Publication 800-20, which defines testing procedures for Triple DES implementations in ECB, CBC, CFB and OFB modes of operation [20]. This publication recommends two classes of tests for verification of the circuit functionality: Known Answer Tests (KATs), and the Monte-Carlo tests. Since, the Known Answer Tests are algorithm specific, we implemented them only for Triple DES. The Monte Carlo test is algorithm independent, so we implemented it for both Triple DES and Rijndael. The operation of this test is shown in Fig. 8. The test consists of 4,000,000 encryptions with keys changed every 10,000 encryptions. The ciphertext block obtained after each sequence of 10,000 encryptions is compared with the corresponding block obtained using software implementation. Software implementations of Triple DES and Rijndael from publicly available Crypto++ 4.1 library were used in our experiments.

The second group of tests was developed based on the principle similar to the Monte-Carlo tests. One megabyte of data is sent to the board for encryption (or decryption), the result is transferred back to the host, and downloaded again to the board as a subsequent part of input. The procedure is repeated 1024 times, which corresponds to encrypting/decrypting a 1 GB stream of data using CBC mode. Only

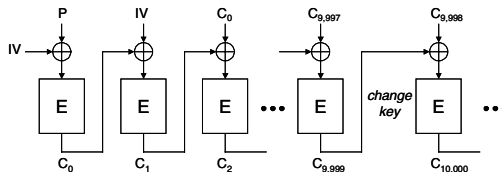


Fig. 8. Monte Carlo Test recommended by NIST in the CBC mode

the last megabyte of output is used for verification, as it depends on all previous input and output blocks. The transfer of data is performed by the DMA unit, so it takes place simultaneously with encryption/decryption. If the test passes, it is repeated at the increased clock frequency. The highest clock frequency at which no single processing error has been detected is considered the maximum clock frequency. In our experiments, this test was repeated 10 times with consistent results in all iterations.

The third group of tests is an extension of the second group. After determining the maximum clock frequency, we measure the amount of time necessary to process 4 GB of data, taking into account the limitations imposed by the 32 bit/33 MHz PCI interface. Since data is transmitted through the PCI interface in both directions (input and output), the maximum encryption/decryption throughput that can be possibly measured using this test is equal to 528 Mbit/s. This is a half of the maximum throughput in the regular operation of the FPGA accelerator, where only input data are transferred from the host to the accelerator card through the PCI interface, and the output is transferred from the FPGA card to the Ethernet daughter card.

6 Results

The results of static timing analysis and experimental testing for Rijndael and Triple DES are shown in Fig. 9.

For Triple DES in the basic iterative architecture, the maximum clock frequency is equal to 72 MHz according to the static analyzer, and 91 MHz according to the experimental testing using the SLAAC-1V board.

For Rijndael in the basic iterative architecture, the results for encryption and decryption are different, with decryption slower than encryption by about 13% in experimental testing. According to the timing analyzer, the maximum clock frequency for the entire circuit is equal to 47 MHz, with the critical path determined by the decryption circuit. In experimental testing, decryption works correctly up to 52 MHz, and encryption up to 60 MHz. However, we do not intend to change the clock frequency on the fly, therefore 52 MHz sets the limit for the entire circuit. The differences between the static timing analysis and experimental testing are caused by conservative assumptions used by the Xilinx static timing analyzer, including the worst case parameters for voltage and temperature prorating.

In Fig. 9b, the maximum throughputs corresponding to the analyzed and experimentally tested clock frequencies are estimated based on the equation:

$$\text{Maximum_Throughput} = (\text{Block_size} / \text{\#Rounds}) \cdot \text{Maximum_Clock_Frequency}. \quad (15)$$

Using formula (15), the maximum throughput of Rijndael in the basic iterative architecture for a 128-bit key is 521 Mbit/s based on the static timing analysis, and 577 Mbit/s based on the experimentally measured clock frequency. This result is expected to be further improved by optimizations of placement and routing. Taking into account our result, parallel processing of only two streams of data should be sufficient to obtain the speed over 1 Gbit/s. As a result, one stage of additional registers, R2a-c, was added to the basic iterative architecture in the extended

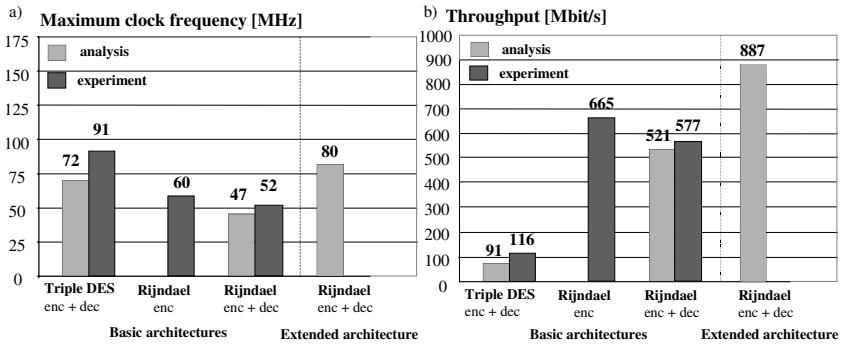


Fig. 9. Results of the static timing analysis and experimental testing for Rijndael and Triple DES a) maximum clock frequency, b) corresponding throughput

architecture as shown in Fig. 3. At this moment, we have been able to obtain a throughput of 887 Mbit/s for this extended pipelined architecture. Nevertheless, further logic and routing optimizations are expected to improve this throughput over 1 Gbit/s without the need of introducing any additional pipeline stages.

The worst-case throughput of Triple DES in the basic iterative architecture is 91 Mbit/s based on the static timing analysis, and 116 Mbit/s based on the experimentally measured maximum clock frequency, which translates to the 27% speed-up in experiment. Sixteen independent streams of data processed simultaneously should easily exceed 1 Gbit/s, leading to the extended architecture shown in Fig. 7.

The actual encryption and decryption throughputs, taking into account the limitations imposed by the PCI interface were measured using the third group of tests described in Section 5. The actual throughputs for DES, were equal to 102 Mbit/s for encryption, and 108 Mbit/s for decryption. The experimentally measured throughput for Rijndael was equal to 404 Mbit/s, and was the same independently of the key size, which means that this throughput was limited by the PCI interface. It should be noted that during the regular operation of the card, when no output is transferred back to the host memory, this throughput can be easily doubled and reach at least 808 Mbit/s.

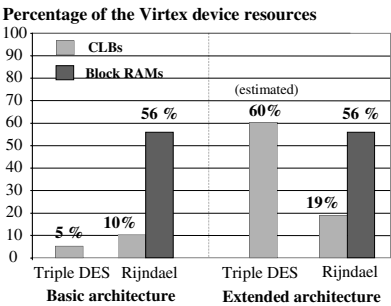


Fig. 10. Percentage of the Virtex device resources devoted to the encryption circuits

The total percentage of the FPGA resources used for the *basic* iterative architectures of Rijndael and Triple DES is 15% of CLB slices and 56% of BlockRAMs. The *extended* architectures of both ciphers, capable of operating over 1 Gbit/s, will take approximately 80% of CLB slices, and 56% of Block SelectRAMs. Only one Virtex XCV-1000 FPGA is necessary to assure the throughput of both ciphers in excess of 1 Gbit/s. Using two additional Virtex devices, and more complex architectures, the encryption throughput in excess of 3 Gbit/s can be accomplished. Our 64-bit/66 MHz PCI module will support this bandwidth.

7 Related Work

Several research groups developed VHDL implementations of Rijndael in Xilinx FPGAs [3, 6, 7, 12, 14], and Altera FPDs [8, 9, 10, 19]. A survey and relative comparison of results from various groups is given in [13]. All major results described in the aforementioned papers are based on the static timing analysis and simulation, and have not yet been confirmed experimentally.

The first attempt to validate the simulation speed of Rijndael through experimental testing is described in [9]. The test was performed using especially developed PCI card. Nevertheless, since the operation of the system appeared to be limited by the PCI controller, no numerical results of the experimental tests were reported in the paper.

As a result, our paper is the first one that describes the successful experimental testing of Rijndael and directly compares the experimental results with simulation.

8 Summary and Possible Extensions

The IPSec-compliant encryption/decryption units of the new Advanced Encryption Standard - Rijndael and the older encryption standard Triple DES have been developed and tested experimentally. Both units support the Cipher Block Chaining mode. Our experiment demonstrated up to 27% differences between the results obtained from testing and results of the static timing analysis based on Xilinx tools. These differences confirmed that the results based on the static analyzer should be treated only as the worst-case estimates.

The experimental procedure demonstrated that the total encryption and decryption throughput of Rijndael and Triple DES in excess of 1 Gbit/s can be achieved using a single FPGA device Virtex 1000. Only up to 80% of resources of this single FPGA device are required by all cryptographic modules. The throughput in excess of 3 Gbit/s can be accomplished by using two remaining FPGA devices present on the SLAAC-IV accelerator board. The alternative extensions include the implementation and experimental testing of other security transformations of IPSec, such as HMAC and the Internet Key Exchange protocol.

References

1. Advanced Encryption Standard Development Effort. <http://www.nist.gov/aes>
2. Blaze M., Diffie W., Rivest R., Schneier B., Shimomura T., Thompson E., and Wiener M.: Minimal Key Lengths for Symmetric Ciphers to Provide Adequate Commercial Security <http://www.counterpane.com/keylength.html>
3. Chodowiec P., Khuon P., Gaj K.: Fast Implementations of Secret-Key Block Ciphers Using Mixed Inner- and Outer-Round Pipelining. Proc. ACM/SIGDA Ninth International Symposium on Field Programmable Gate Arrays, FPGA'01, Monterey, Feb. 2001, 94-102
4. Davida G. I. and Dancs F. B.: A crypto-engine, Proc. CRYPTO 87, (1987) 257-268
5. Daemen J. and Rijmen V.: Rijndael: Algorithm Specification. <http://csrc.nist.gov/encryption/aes/rijndael/>
6. Dandalis A., Prasanna V. K., Rolim J. D.: A Comparative Study of Performance of AES Final Candidates Using FPGAs. Proc. Cryptographic Hardware and Embedded Systems Workshop, CHES 2000, Worcester, MA, Aug 17-18, 2000
7. Elbirt A. J., Yip W., Chetwynd B., Paar C.: An FPGA implementation and Performance Evaluation of the AES Block Cipher Candidate Algorithm Finalists. Proc. 3rd Advanced Encryption Standard (AES) Candidate Conference, New York, April 13-14, 2000
8. Fischer V.: Realization of the Round 2 AES Candidates Using Altera FPGA. Submitted for 3rd Advanced Encryption Standard (AES) Candidate Conference, New York, April 13-14, 2000; <http://csrc.nist.gov/encryption/aes/round2/conf3/aes3papers.html>
9. Fisher V.: Realization of the RIJNDAEL Cipher in Field Programmable Devices. Proc. of DCIS 2000, Montpellier, Nov. 2000, 312-317
10. Fisher V., Drutarovský M.: Two methods of Rijndael implementation in reconfigurable hardware. Proc. of CHES 2001, Paris, 2001
11. Frankel S., Kelly S., Glenn R.: The AES Cipher Algorithm and Its Use with IPsec. Network Working Group Internet Draft, November 2000, (work in progress) available at <http://ietf.org/html.charters/ipsec-charter.html>
12. Gaj K., Chodowiec P.: Comparison of the Hardware Performance of the AES Candidates Using Reconfigurable Hardware. Proc. 3rd Advanced Encryption Standard (AES) Candidate Conference, New York, April 13-14, 2000
13. Gaj K. and Chodowiec P.: Hardware performance of the AES finalists survey and analysis of results, Technical Report available at <http://ece.gmu.edu/crypto/publications.htm>
14. Gaj K. and Chodowiec P.: Fast Implementation and Fair Comparison of the Final Candidates for Advanced Encryption Standard Using Field Programmable Gate Arrays, Proc. RSA Security Conference - Cryptographer's Track, April 2001
15. IP Security Protocol (ipsec) Charter - Latest RFCs and Internet Drafts for IPsec, <http://ietf.org/html.charters/ipsec-charter.html>
16. Jones M., Athanas P. *et al.*: Implementing an API for Distributed Adaptive Computing Systems. IEEE Workshop on Field-Programmable Custom Computing Machines, Napa Valley, CA, Apr. 1999, 222-230
16. Lipmaa H., Rogaway P., Wagner D.: CTR-Mode Encryption, Public Workshop on Symmetric Key Block Cipher Modes of Operation. Oct. 2000, Baltimore, MD, <http://csrc.nist.gov/encryption/modes/workshop1/>
18. Modes of Operation. <http://csrc.nist.gov/encryption/modes/>
19. Mroczkowski P.: Implementation of the Block Cipher Rijndael Using Altera FPGA. Public Comments on AES Candidate Algorithms - Round 2. <http://csrc.nist.gov/encryption/aes/round2/pubcmnts.htm>.
20. NIST Special Publication 800-20, Modes of Operation Validation System for the Triple Data Encryption Algorithm, National Institute of Standards and Technology (2000)
21. Secure Hash Standard Home Page. <http://csrc.nist.gov/cryptval/shs.html>
22. Smith R. E.: Internet Cryptography, Addison-Wesley (1997)

Elliptic Curve Arithmetic Using SIMD

Kazumaro Aoki^{1*}, Fumitaka Hoshino², Tetsutaro Kobayashi², and
Hiroaki Oguro²

¹ NTT Communications

² NTT Information Sharing Platform Laboratories, NTT Corporation
{maro,fhoshino,kotetsu,oguro}@isl.ntt.co.jp

Abstract. Focusing on servers that process many signatures or ciphertexts, this paper proposes two techniques for parallel computing with SIMD, which significantly enhances the speed of elliptic curve scalar multiplication. We also evaluate one of them based on a real implementation on a Pentium III, which incorporates the SIMD architecture. The results show that the proposed method is about 4.4 times faster than the conventional method.

1 Introduction

Elliptic curve cryptosystems have become an essential technology since they provide many advantages such as a short key length [1] and fast computation. The use of elliptic curves in cryptography was suggested independently by Miller [2] and Koblitz [3] in 1985. The elliptic curve cryptosystem is an elliptic curve analog of a discrete logarithm based scheme. Elliptic curves can be defined over any finite field \mathbb{F}_q . If q is prime then \mathbb{F}_q is a ‘prime field.’ If q is of the form 2^m then \mathbb{F}_q is a ‘binary field.’ Elliptic curves over prime and binary fields have become widely accepted and included in standards approved by accredited organizations such as ANSI, IEEE, and ISO.

Many efficient computational algorithms have been proposed for the elliptic curve cryptosystems [4,5]. Several kinds of smart cards have a co-processor which accelerate elliptic curve calculations. Moreover, the recent results shows that we can implement elliptic curve on a smart card without co-processor [6].

As many smart cards are widely distributed, elliptic curve implementations must be accelerated especially for server systems that process a large number of signatures or ciphertexts such as electronic voting, electronic cash, time stamping, and certificate authentication. Koyama and Tsuruoka [7] studied a parallel multiplication environment for elliptic curve arithmetic. Recently, Smart [8] proposed the parallel implementation for a special class of elliptic curves called “Hessian form,” namely those with a point of order three.

This paper discusses parallel implementations of elliptic curves over both prime and binary fields. We describe the implementations over prime fields in Sect. 2 and those over binary fields in Sect. 3.

* This work was done while the author was in NTT Information Sharing Platform Laboratories.

2 Parallel Elliptic Curve Arithmetic

2.1 SIMD

The CPU word size has increased year by year from 32 to 64 and now to 128 bits. Most of these CPUs have the Single Instruction Multiple Data (SIMD) architecture which means that multiple data sets can be processed in parallel. This architecture becomes particularly important as the clock frequency of a CPU approaches the limit. Now, many processors incorporate the SIMD architecture such as the Pentium III has MMX and SSE, SPARC has VIS, and the PowerPC has AltiVec.

Lipmaa [9] presented results on a secret key cryptosystem. Dixon and Lenstra [10] presented results on a factoring. The papers showed that the SIMD characteristic of parallel execution is useful in the quick encryption of a secret key cipher and a factoring. However, the characteristics of SIMD, already explained before, does not depend on secret key cipher properties, but also that it can perform fast public key cryptosystems. This paper utilizes those characteristic to optimize elliptic curve implementations.

It is efficient to use SIMD technology if every operation can be executed simultaneously. However, there is a problem with the data dependency chain. Operations that SIMD processes must be independent of each other.

Consider the following example. When we compute $A + B + C + D$, first, we compute $U \leftarrow A + B$. Next, we compute $V \leftarrow U + C$. Finally, we compute $R \leftarrow V + D$. Then, we have the summation $A + B + C + D$. We cannot apply SIMD to such an algorithm. Changing the order of the computation solves the problem in this case. First, we compute $S \leftarrow A + B$ and $T \leftarrow C + D$ simultaneously. Next, we compute $R \leftarrow S + T$. Then, we have the summation $A + B + C + D$ using the SIMD processor.

We propose semi-optimal methods using SIMD to compute elliptic curve addition and doubling over \mathbb{F}_q , where $\text{char } \mathbb{F}_q > 3$.

2.2 Elliptic Curve Arithmetic over \mathbb{F}_q , $\text{char } \mathbb{F}_q > 3$

A non-supersingular elliptic curve defined over the finite field, \mathbb{F}_q ; $\text{char } \mathbb{F}_q > 3$ is given by (I).

$$E : y^2 = x^3 + ax + b \quad (a, b \in \mathbb{F}_q, 4a^3 + 27b^2 \neq 0) \quad (1)$$

We denote the group of \mathbb{F}_q -rational points (x, y) on E together with a ‘point at infinity’ \mathcal{O} as $E(\mathbb{F}_q)$.

In Jacobian coordinates, we set $(x, y) = (X/Z^2, Y/Z^3)$, giving the equation

$$E_J : Y^2 = X^3 + aXZ^4 + bZ^6, \quad (2)$$

and represent rational points on E by (X, Y, Z) .

If a scalar multiplication is implemented by the binary method [11, pp.614–615], we can assume that the addend of elliptic curve additions is normalized, i.e.,

$Z = 1$, since the addend is the input of the method. Even if a scalar multiplication is implemented by the Window method [11, pp.615-616] [8], we can also assume that the addend of the elliptic curve additions is normalized according to [4]. In this case, elliptic curve addition $(X_2, Y_2, Z_2) \leftarrow (X_0, Y_0, Z_0) + (X_1, Y_1, 1)$ and elliptic curve doubling $(X_2, Y_2, Z_2) \leftarrow 2(X_0, Y_0, Z_0)$ are defined in Figure 1.

Addition formulae	Doubling formulae
$X_2 \leftarrow -H^3 - 2U_1H^2 + r^2$ $Y_2 \leftarrow -S_1H^3 + r(U_1H^2 - X_2)$ $Z_2 \leftarrow Z_0H$ where $U_1 = X_1Z_0^2$ $S_1 = Y_1Z_0^3$ $H = X_0 - U_1$ $r = S_1 - Y_0$	$X_2 \leftarrow T$ $Y_2 \leftarrow -8Y_0^4 + M(S - T)$ $Z_2 \leftarrow 2Y_0Z_0$ where $S = 4X_0Y_0^2$ $M = 3X_0^2 + aZ_0^4$ $T = -2S + M^2$

Fig. 1. Elliptic curve addition and doubling formulae over \mathbb{F}_q , $\text{char } \mathbb{F}_q > 3$.

2.3 Elliptic Curve Arithmetic with Parallel Multiplication

Currently, more than two calculations can be performed in parallel using SIMD instructions. However, the following problems occur.

- Calculations that depend on the results of one of the other calculations cannot be computed simultaneously.
- Calculations of two elements in the same register require data place-adjustment operations. The cost cannot be ignored.

Because of these characteristics, it is difficult to implement a field multiplication using SIMD. On the other hand, we need to perform several field multiplications to establish an elliptic curve addition or doubling. If we can perform two or more independent field multiplications of an elliptic curve addition or doubling in parallel, these SIMD problems can be resolved because each parallel calculator of the SIMD can perform independent field multiplications.

Koyama and Tsuruoka [7] studied a parallel multiplication environment for elliptic curve arithmetic. However, they only estimated the number of parallel processes and did not show how to compute an elliptic curve arithmetic using multiple field multipliers in a practical situation. We carefully observed field multiplication calculations in elliptic curve arithmetic and found that we can compute field multiplication calculations in parallel without many multiplier stalls for an elliptic curve, whose elements are represented as a projective coordinate.

We propose an algorithm of the elliptic curve addition and doubling of Jacobian coordinates (Figure 1) using two field multipliers is presented in Figures 2

¹ Sometimes referred as the k -ary method for an exponentiation.

and [3]. Note that field additions and subtractions are partially omitted. The asterisk (*) in the figures represents squaring.

We propose a modified Jacobian coordinate system that is suitable for parallel operations. In modified Jacobian coordinates, we represent a rational point on E by (X, Y, Z, Z^2) instead of (X, Y, Z) of the original Jacobian coordinates. Though there were similar modified representation of Jacobian coordinates to decrease the number of field multiplication for elliptic curve arithmetic [4], our viewpoint is new, because we can compute elliptic curve addition and doubling with the same number of field multiplications but with fewer steps in the modified Jacobian coordinates than in the original Jacobian coordinates. We also show an algorithm of the elliptic curve addition and doubling in the modified Jacobian coordinates using three field multipliers in Figures 4 and 5. Note that field additions and subtractions are partially omitted.

We can compute the elliptic curve addition and doubling in Figure 1 with 11 and 9 steps, respectively, by using one field multiplier. On the other hand, [7] shows the theoretical bound of the parallelization. To establish the elliptic curve addition and doubling, we need at least 3 steps even if we use 29 parallel field multipliers. Recently, Smart [8] proposed the parallel implementation of a special class of elliptic curves called “Hessian form,” namely those with a point of order three. He showed that the elliptic curve addition and doubling derived from Hessian form can be computed with 4 and 3 steps, respectively, by using three parallel field multipliers.

Step	Multiplier #1	Multiplier #2
1	Z_0^2	$Y_1 \cdot Z_0$
2	$U_1 \leftarrow X_1 \cdot Z_0^2$	$S_1 \leftarrow Y_1 Z_0 \cdot Z_0^2$
3*	H^2	r^2
4	$H \cdot H^2$	$U_1 \cdot H^2$
5	$S_1 \cdot H^3$	$Z_2 \leftarrow Z_0 \cdot H$
6	$r \cdot (U_1 H^2 - X_2)$	—

Fig. 2. Parallel elliptic curve addition for Jacobian coordinate

Step	Multiplier #1	Multiplier #2
1*	Y_0^2	Z_0^2
2*	X_0^2	$(Z_0^2)^2$
3*	$(Y_0^2)^2$	M^2
4	$S \leftarrow 4X_0 \cdot Y_0^2$	$Z_2 \leftarrow 2Y_0 \cdot Z_0$
5	$M \cdot (S - T)$	—

Fig. 3. Parallel elliptic curve doubling for Jacobian coordinate

Thus, we believe that the proposed method is semi-optimal because Figures 2 and 3 compute elliptic curve addition and doubling with 6 and 5 steps, respectively, by using two parallel field multipliers and Figures 4 and 5 compute them with 4 and 3 steps, respectively, by using three parallel field multipliers. Though it can use a general elliptic curve parameter, it is the same as efficient as parallel algorithm for special elliptic curves 8. Their number of field multiplication is the same as the case of one field multiplier and the probability is only 10% or less that a multiplier sleeps.

Step	Multiplier #1	Multiplier #2	Multiplier #3
1	$U_1 \leftarrow X_1 \cdot Z_0^2$	$Z_0 \cdot Z_0^2$	—
2	H^2	$S_1 \leftarrow Y_1 \cdot Z_0^3$	$Z_2 \leftarrow Z_0 \cdot H$
3	$U_1 \cdot H^2$	$H \cdot H^2$	r^2
4	$S_1 \cdot H^3$	$r \cdot (U_1 H^2 - X_2)$	Z_2^2

Fig. 4. Parallel elliptic curve addition for modified Jacobian coordinate

Step	Multiplier #1	Multiplier #2	Multiplier #3
1*	Y_0^2	X_0^2	$(Z_0^2)^2$
2	M^2	$S \leftarrow 4X_0 \cdot Y_0^2$	$Z_2 \leftarrow 2Y_0 \cdot Z_0$
3	$(Y_0^2)^2$	$M \cdot (S - T)$	Z_2^2

Fig. 5. Parallel elliptic curve doubling for modified Jacobian coordinate

3 Bitslice Implementation

Bitslice implementation is a technique that regards the w -bit CPU as a w -parallel 1 bit CPU, where w is the word size of the CPU. This is also a kind of SIMD. This technique has been applied to secret key cryptosystems [12][13].

But there has been no application to public key cryptosystem because the usual public key algorithm requires conditional branches. This complicates the bitslice implementation. This paper first applies the bitslice technique to public key cryptosystem and apply it to elliptic curve cryptosystems defined over \mathbb{F}_{2^m} . When independent w elliptic curve operations are simultaneously performed, each of the word bit is allocated to each curve. We can avoid conditional branches by introducing a new technique, which realizes a conditional move, for the bitslice implementation.

The organization of this section is as follows. Subsection 3.1 describes the data structure of the bitslice implementation. Subsection 3.2 describes the conditional move techniques. Subsections 3.3, 3.5, and 3.6 describe the performance in terms of efficiency.

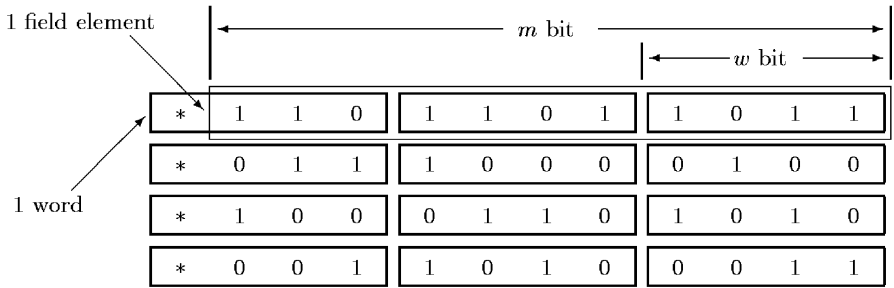


Fig. 6. Data representation of non-bitslice implementation, ($m = 11, w = 4$)

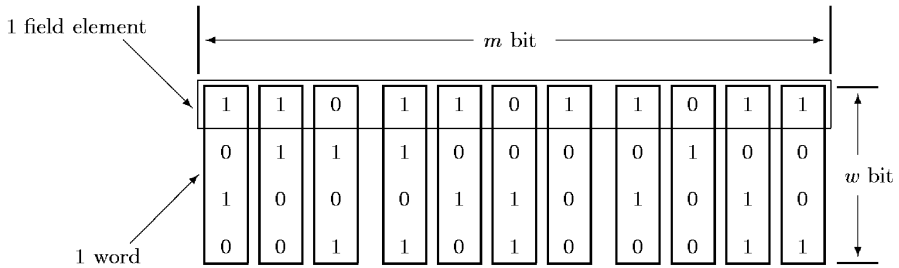


Fig. 7. Data representation of bitslice implementation, ($m = 11, w = 4$)

3.1 Data Structure

In this subsection, we describe how to represent \mathbb{F}_{2^m} arithmetic by using the bitslice implementation.

We assume parallel processing in w , i.e., the word size of the processor, with independent elements over \mathbb{F}_{2^m} . Using a non-bitslice implementation, w elements in \mathbb{F}_{2^m} are represented in Figure 6, where one row is divided into several words and one word is denoted by one rectangle. We represent them in Figure 7 when using the bitslice implementation. That is, we denote the i -th bit of the j -th element in \mathbb{F}_{2^m} as $e_{(i,j)}$,

$$e_i = \sum_{j=0}^{w-1} e_{(i,j)} 2^j. \quad (0 \leq i < m)$$

w elements in \mathbb{F}_{2^m} can be represented by m word data $e_i (0 \leq i < m)$. The e_i is a bitslice representation.

The advantage of the bitslice implementation is that bitwise operation becomes easy, because each bit of data is contained in a different register. On the other side, bitslice implementation is inefficient if there are less data to be processed than w .

3.2 Conditional Move

It is difficult to realize a conditional branch when using the bitslice implementation, because one register has w independent data and each of w branches independently. For example, when we use a binary method to compute the elliptic curve scalar multiplication by k , we must establish a branch corresponding to each bit of k . In this subsection, the basic idea of the conditional move in the bitslice implementation is shown and the problem of a binary method without a conditional branch is solved.

Let the bold face characters denote the vector of w data, e.g., \mathbf{A} denotes the vector of w rational points $(A(0), A(1), \dots, A(w-1))$ and \mathbf{k} denotes the vector of w independent scalars $(k(0), k(1), \dots, k(w-1))$. We define the two-case conditional move $\text{cmov}_2(\mathbf{k}; \mathbf{A}, \mathbf{B})$ such that

$$\text{cmov}_2(\mathbf{k}; \mathbf{A}, \mathbf{B}) := \mathbf{T},$$

$$\text{where } T(i) = \begin{cases} A(i) & \text{if } k(i) = 1, \\ B(i) & \text{if } k(i) = 0, \end{cases}$$

$$\text{for } 0 \leq i < w.$$

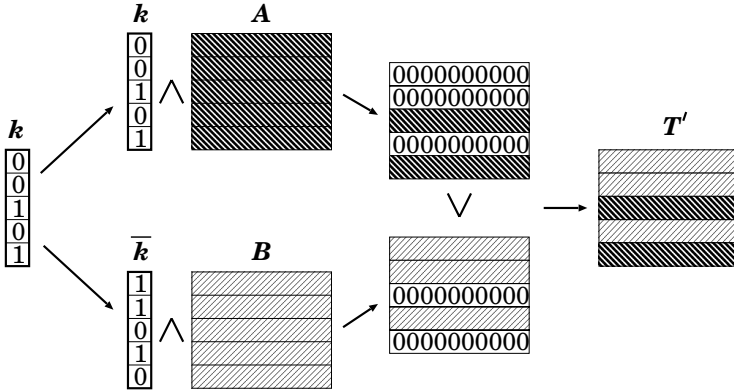


Fig. 8. Example of bitslice conditional move realization for bitslice implementation

The basic idea of the conditional move realization for bitslice implementation is shown in Figure 8. The figure shows that we can compute cmov_2 using the fact that

$$\begin{aligned} \mathbf{T}' &= (\mathbf{k} \wedge \mathbf{A}) \vee (\bar{\mathbf{k}} \wedge \mathbf{B}) \\ &= \text{cmov}_2(\mathbf{k}; \mathbf{A}, \mathbf{B}) \end{aligned}$$

where $\bar{\mathbf{k}}$ is a bitwise complementation of \mathbf{k} . \mathbf{T}' is equivalent to output \mathbf{T} of cmov_2 using conditional branches for each point.

The cost of cmov_2 is nearly equal to three field additions, which is negligible compared to an elliptic curve addition.

3.3 The Window Method for Bitslice Implementation

The conditional move enables us to use the window method that is faster than a binary method. Although we have more complicated conditional branches in the conventional window method, we can replace all of them using conditional moves for multiple elements.

We show the window method for bitslice implementation by using a four-case conditional move cmov_4 in Figure 9, where the window size equals 2. We can construct cmov_4 by using three cmov_2 as

$$\text{cmov}_4(\mathbf{k}_1, \mathbf{k}_0; \mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}) := \text{cmov}_2(\mathbf{k}_1; \text{cmov}_2(\mathbf{k}_0; \mathbf{A}, \mathbf{B}), \text{cmov}_2(\mathbf{k}_0; \mathbf{C}, \mathbf{D})).$$

Since cmov_s , ($s > 2$) can be constructed similarly for any s , we can choose any window size for the window method in the bitslice implementation. Note that \mathcal{O} denotes $(\mathcal{O}, \mathcal{O}, \dots, \mathcal{O})$ in Figure 9.

Input : \mathbf{P}, \mathbf{k}_j , $(0 \leq j < t, t \text{ is even})$
 Output : $\sum_{j=0}^{t-1} 2^j k_j(i) P(i)$, $(0 \leq i < w)$

1. Compute $\mathbf{P}_2 \leftarrow 2\mathbf{P}$, $\mathbf{P}_3 \leftarrow 3\mathbf{P}$, $\mathbf{T} \leftarrow \mathcal{O}$.
2. For $j \leftarrow t-1$ down to 1 decremented by 2
 - 2.1 $\mathbf{T} \leftarrow 4\mathbf{T}$.
 - 2.2 $\mathbf{U} \leftarrow \text{cmov}_4(\mathbf{k}_j, \mathbf{k}_{j-1}; \mathbf{P}_3, \mathbf{P}_2, \mathbf{P}, \mathcal{O})$.
 - 2.3 $\mathbf{T} \leftarrow \mathbf{T} + \mathbf{U}$.
3. Return \mathbf{T} .

Fig. 9. Window method for bitslice implementation (window size = 2)

The numbers of computations for the non-bitslice implementation and bitslice implementation are shown in Table 1, where l is the size of window. A_b and A_n denote the cost of elliptic curve addition using the bitslice and non-bitslice implementations, respectively. Also D_b and D_n denote the cost of elliptic curve doubling using the bitslice and the non-bitslice implementation, respectively. We give the detailed evaluation of A_b and A_n in Appendix. We omit the detail about D_b and D_n because we can compute scalar multiplication without elliptic curve doublings using the parameter K-163 [14].

We can assume a window size to be at least $l = 4$, in the conventional implementation of the window method for a secure curve. The difference between the implementation of a conditional branch and a conditional move

$$\frac{\left(\left(\frac{t}{l} + 2^l - 2 \right) A_b + t D_b \right)}{\left(\left(\frac{(2^l - 1)t}{2^l} + 2^l - 2 \right) A_n + t D_n \right)} - 1$$

² We must use conditional branch for elliptic curve addition with \mathcal{O} . We can also use cmov_2 to solve the problem.

Table 1. Number of Computations for Bitslice and Non-bitslice Scalar Multiplications

	Bitslice (with conditional move)	Non-bitslice (with conditional branch)
Binary method	$w(tA_b + tD_b)$	$\frac{1}{2}tA_n + tD_n$
Window method	$w\left(\left(\frac{t}{l} + 2^l - 2\right)A_b + tD_b\right)$	$\left(\frac{(2^l - 1)t}{2^l} + 2^l - 2\right)A_n + tD_n$

is only 6.7% at most, assuming usual situation as $0 \leq D_b \leq D_n$, $0 \leq A_b \leq A_n$ and $t > 0$.

3.4 Efficiency

In this section, we describe the performance in terms of efficiency, assuming the following.

- There are two typical representation of \mathbb{F}_{2^m} , namely in a normal basis and a polynomial basis. We select the polynomial basis because it is efficient if we use the Karatsuba algorithm [15, pp.294-302] to compute field multiplication.
- We can apply the Karatsuba algorithm recursively to both, the bitslice and the non-bitslice implementations until single precision multiplication. Because single precision is w -bit for the non-bitslice and 1-bit for the bitslice implementation, the depth of recursions of the Karatsuba algorithm differs between bitslice and non-bitslice implementations.
- We use Itoh-Tsujii inversion algorithm [16] for the bitslice implementation, and almost inverse algorithm [17] for the non-bitslice implementation.
- We assume that rational points are in $E(\mathbb{F}_{2^m})$, $m = 163$, and the CPU word size, w , is 64.
- We show the number of computations per scalar multiplication in Table 2, because the bitslice implementation independently computes the scalar multiplication w times at once. We assume $s = x = a = 1$ cycle, where a , x and s denote the costs of one word operations **AND**, **XOR** and **SHIFT**, respectively.
- We ignore the costs to convert the non-bitslice representation and bitslice representation because they are less compared to one field multiplication in \mathbb{F}_{2^m} .
- We use the NIST recommended elliptic curve parameter [14] over \mathbb{F}_{2^m} . Digits written in a courier font denote a hexadecimal number.

[NIST-recommended elliptic curve K-163]

$E : y^2 + xy = x^3 + ax^2 + b$, $a = 1, b = 1$

The field polynomial: $p(t) = t^{163} + t^7 + t^6 + t^3 + 1$

The base point order:

$r = 4$ 00000000 00000000 00020108 a2e0cc0d 99f8a5ef

The base point $G = (G_x, G_y)$

$G_x = 2$ fe13c053 7bbc11ac aa07d793 de4e6d5e 5c94eee8

$G_y = 2$ 89070fb0 5d38ff58 321f2e80 0536d538 ccdaa3d9

3.5 Comparison Based on an Ideal Model

In this subsection, we compare bitslice and non-bitslice implementations based on an ideal model. The numbers are based on the tables in Appendix.

Table 2. Number of Computations for One Elliptic Curve Addition ($w = 64, m = 163$)

Coordinate	Bitslice	Non-bitslice
Affine	4302	5245
Projective [5]	2690	16790

(cycles)

Table 2 shows that elliptic curve addition by the projective coordinate is faster when we use the bitslice implementation, and by the affine coordinate is faster when we use the non-bitslice implementation. In comparison with the bitslice implementation in the projective coordinate and the non-bitslice implementation in the affine coordinate, the bitslice one is about twice as fast as the non-bitslice one.

3.6 Comparison by Implementation

In this subsection, we compare the bitslice and the non-bitslice implementations by real implementation on a Pentium III PC. The specifications for the computer we used are

CPU : Pentium III 850MHz
Memory : 128MB

We show the results of the implementation in Table 3. It shows that the bitslice implementation is 4.4 times faster than non-bitslice.

Table 3. Time of Computations for an Elliptic Curve Scalar Multiplication

Coordinate	Bitslice (with conditional move)	Non-bitslice (with conditional branch)
Affine	552.9	1444
Projective [5]	326.8	1918

($m = 163, \mu\text{sec.}$, Pentium III 850 MHz, Memory 128MB)

4 Conclusion

With the focus of processing many signatures or ciphertexts, we proposed two techniques, the “parallel elliptic curve algorithm” and use of “bitslice implementation,” to accelerate elliptic curve arithmetic over prime and binary fields, respectively.

The parallel elliptic curve addition and doubling methods for prime field enable two or three parallel computations. By the technique, we can utilize SIMD architecture effectively. Their number of computation is the same as the case of one field multiplier and the probability is only 10% or less that a multiplier sleeps.

The bitslice implementation accelerates field arithmetic over a binary field. Though the usual public key algorithm requires conditional branches, we first applied the bitslice technique to public key cryptosystem, avoiding conditional branches by introducing a new technique, which realizes a “conditional move, for the bitslice implementation.” The performance depends on the circumstances. Our implementation shows that the speed of the bitslice implementation is 4.4 times faster than the conventional non-bitslice implementation on a Pentium III.

References

1. Lenstra, A.K., Verheu, E.R.: Selecting cryptographic key sizes. In Imai, H., Zheng, Y., eds.: Public Key Cryptography — Third International Workshop on Practice and Theory in Public Key Cryptosystems, PKC 2000. Volume 1751 of Lecture Notes in Computer Science. Springer-Verlag, Berlin, Heidelberg, New York (2000) 446–465
2. Miller, V.S.: Use of elliptic curves in cryptography. In Williams, H.C., ed.: Advances in Cryptology — CRYPTO’85. Volume 218 of Lecture Notes in Computer Science. Springer-Verlag, Berlin, Heidelberg, New York (1986) 417–426
3. Koblitz, N.: Elliptic curve cryptosystems. *Mathematics of Computation* **48** (1987) 203–209
4. Cohen, H., Miyaji, A., Ono, T.: Efficient elliptic curve exponentiation using mixed coordinates. In Ohta, K., Pei, D., eds.: Advances in Cryptology — ASIACRYPT’98. Volume 1514 of Lecture Notes in Computer Science. Springer-Verlag, Berlin, Heidelberg, New York (1998) 51–65
5. Lopez, J., Dahab, R.: Improved algorithms for elliptic curve arithmetic in $gf(2^n)$. In Tavares, S., Meijer, H., eds.: Selected Areas in Cryptography — 5th Annual International Workshop, SAC’98. Volume 1556 of Lecture Notes in Computer Science., Berlin, Heidelberg, New York, Springer-Verlag (1999) 210–212
6. Woodbury, A., Bailey, D., Paar, C.: Elliptic curve cryptography on smart cards without coprocessors. In: the Fourth Smart Card Research and Advanced Applications (CARDIS 2000) Conference. CADIS’2000, Bristol, UK (2000)
7. Koyama, K., Tsuruoka, Y.: Speeding up elliptic curve cryptosystems by using a signed binary windows method. In Brickell, E.F., ed.: Advances in Cryptology — CRYPTO’92. Volume 740 of Lecture Notes in Computer Science. Springer-Verlag, Berlin, Heidelberg, New York (1993) 345–357
8. Smart, N.P.: The hessian form of an elliptic curve. In: Preproceedings of Cryptographic Hardware and Embedded Systems. CHES2001 (2001) 121–128
9. Lipmaa, H.: Idea: A cipher for multimedia architectures? In Tavares, S., Meijer, H., eds.: Selected Areas in Cryptography — 5th Annual International Workshop, SAC’98. Volume 1556 of Lecture Notes in Computer Science., Berlin, Heidelberg, New York, Springer-Verlag (1999) 248–263
10. Dixon, B., Lenstra, A.K.: Factoring integers using simd sieves. In Helleseht, T., ed.: Advances in Cryptology — EUROCRYPT’93. Volume 765 of Lecture Notes in Computer Science. Springer-Verlag, Berlin, Heidelberg, New York (1993) 28–39

11. Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: Handbook of applied cryptography. CRC Press (1997)
12. Biham, E.: A fast new des implementation in software. In Biham, E., ed.: Fast Software Encryption — 4th International Workshop, FSE'97. Volume 1267 of Lecture Notes in Computer Science. Springer-Verlag, Berlin, Heidelberg, New York (1997) 260–272
13. Nakajima, J., Matsui, M.: Fast software implementations of misty1 on alpha processors. IEICE Transactions Fundamentals of Electronics, Communications and Computer Sciences (Japan) **E82-A** (1999) 107–116
14. NIST: Recommended elliptic curves for federal government use (1999) (available at <http://csrc.nist.gov/csrc/fedstandards.html>).
15. Knuth, D.E.: Seminumerical Algorithms. Third edn. Volume 2 of The Art of Computer Programming. Addison Wesley (1997)
16. Itoh, T., Tsujii, S.: A fast algorithm for computing multiplicative inverses in $gf(2^m)$ using normal bases. In: Information and Computation. Volume 78. (1988) 171–177
17. Schroepel, R., Orman, H., O'Malley, S., Sparscheck, O.: Fast key exchange with elliptic curve systems. In Coppersmith, D., ed.: Advances in Cryptology — CRYPTO'95. Volume 963 of Lecture Notes in Computer Science. Springer-Verlag, Berlin, Heidelberg, New York (1995) 43–56

Appendix

In this appendix we show in detail the number of computations for elliptic curve additions using the bitslice and non-bitslice implementations A_b and A_n in Tables 4 and 5.

M , S and I denote the cost of multiplication, squaring, and an inversion of elements in \mathbb{F}_{2^m} . We assume the base of log is 2.

Table 4. Number of Computations for Bitslice Implementation

Coordinate	Operation	Bitslice (per one element)
Affine		$2M + S + I$
	s	0
	x	$\frac{1}{3w} \{26 + 13 \cdot w_H(m-1)\} m^{\log 3} + \frac{1}{2w} tm$
	a	$\frac{1}{9w} \{32 + 16 \cdot w_H(m-1)\} m^{\log 3}$
Projective [5]		$9M + 4S$
	s	0
	x	$\frac{1}{w} \{39m^{\log 3} + 2tm\}$
	a	$\frac{16}{w} m^{\log 3}$

Table 5. Number of Computations for Non-bitslice Implementation

Coordinate	Operation	Non-bitslice
Affine		$2M + S + I$
	s	$2w(\frac{m}{w})^{\log 3} + 2(2m + \log w) \lceil \frac{m}{w} \rceil$
	x	$2(9w - 2)(\frac{m}{w})^{\log 3} + 2(2m + \log w) \lceil \frac{m}{w} \rceil$
	a	$2w(\frac{m}{w})^{\log 3} + 2 \lceil \frac{m}{w} \rceil \log w$
Projective [5]		$9M + 4S$
	s	$9w \cdot (\frac{m}{w})^{\log 3} + 8 \lceil \frac{m}{w} \rceil \log w$
	x	$9(9w - 2)w \cdot (\frac{m}{w})^{\log 3} + 8 \lceil \frac{m}{w} \rceil \log w$
	a	$9w \cdot (\frac{m}{w})^{\log 3} + 8 \lceil \frac{m}{w} \rceil \log w$

On the Hardware Implementation of the 3GPP Confidentiality and Integrity Algorithms

Kostas Marinis¹, Nikos K. Moshopoulos², Fotis Karoubalis², Kiamal Z. Pekmestzi¹

¹ National Technical University of Athens,
Electrical & Computer Engineering Dept.,
9 Heroon Polytehneiou, 15773 Zographou, Athens, Greece
{kmarinis, pekmes}@microlab.ntua.gr

² Atmel Hellas, Multimedia & Communications Group,
34 Ag. Varvaras Ave. & Taxiarchon Str.,
17563 Paleo Faliro, Athens, Greece
{nmoshopoulos, fkaroubalis}@athens.atmel.com

Abstract. This paper discusses the design and implementation of the Confidentiality and Integrity algorithms, which have been standardized by the 3rd Generation Partnership Project. Both algorithms use a modified version of the MISTY scheme, named KASUMI, as a basic cryptographic engine. Various architectural approaches have been examined and implemented in different hardware platforms (FPGAs, ASICs) providing useful information regarding the required area and the design throughput.

1 Introduction

Two algorithms have been standardized by the ETSI/SAGE regarding the 3rd Generation Partnership Program (3GPP) Security of Data: the Confidentiality algorithm (f_8) and the Integrity algorithm (f_9) [1]. The f_8 function is used to protect the confidentiality of user signaling data sent over the radio access link. The f_9 is a Message Authentication Code (MAC) function, used to authenticate the data integrity and data origin of signaling data. Both algorithms employ a modification of the MISTY scheme as their basic cryptographic engine. This modification is called the KASUMI algorithm [2] and its effective hardware implementation is essential for the system.

In this paper we investigate the applicability of KASUMI to hardware, by implementing the algorithm on three different hardware platforms. In section 2, the KASUMI algorithm is presented, along with the architectural approaches examined in this paper. The actual implementation of the KASUMI schemes is described in section 3, taking into account various special issues regarding the target technologies, namely the Xilinx and Altera FPGAs and the Atmel 0.25 μm ASIC process. The KASUMI algorithm is then used as the basic building block for the implementation of the f_8 and f_9 functions in section 4. In the same section, a serial architecture for the confidentiality and integrity algorithms is presented, which meets the hardware resources requirements described in [3].

2 The KASUMI Algorithm

The KASUMI algorithm is based on MISTY1 [4], a block cipher algorithm designed to be provably secure against differential and linear cryptanalysis. MISTY's architecture was an ideal basis for an algorithm that would fulfill the basic requirements imposed by the 3GPP: to provide a high level of security within the 3GPP context and to meet demanding hardware implementation requirements — in particular, allow a low power, low gate count implementation in hardware. Therefore, several enhancements were performed in order to evolve to KASUMI, an algorithm superior in terms of hardware and security.

KASUMI is an eight round Feistel block cipher, producing a 64-bit output from a 64-bit input under the control of a 128-bit key. The block diagram of Kasumi is depicted in Fig. 1. [2]

The 64-bit input I is divided into two 32-bit strings $L_0 = I[63:32]$ and $R_0 = I[31:0]$. The output strings of each KASUMI round function $f_i(\cdot)$ are produced from the following equation:

$$R_i = L_{i-1}, L_i = R_{i-1} \oplus f_i(L_{i-1}, RK_i), 1 \leq i \leq 8. \quad (1)$$

The result *OUTPUT* is equal to the 64-bit string derived from the concatenation of L_8 (upper part) and R_8 (lower part) delivered at the end of the eighth round.

The function $f_i(\cdot)$, as depicted in Fig. 1, receives a 32-bit input I and returns a 32-bit output O under the control of a round key RK_i . This function is constructed from two sub-functions, namely FL and FO. These sub-functions utilize different parts of RK_i , therefore the round key can be divided into three sub-keys (KL_i , KO_i , KI_i). The $f_i(\cdot)$ function has two different forms depending on whether it is an even round or an odd round. For rounds 1,3,5 and 7 we define:

$$f_i(I, RK_i) = FO(FL(I, KL_i), KO_i, KI_i). \quad (2)$$

and for rounds 2,4,6 and 8 we define:

$$f_i(I, K_i) = FL(FO(I, KO_i, KI_i), KL_i). \quad (3)$$

As can be observed from equations (2) and (3), the *FL* sub-function comprises KL_i sub-key while *FO* comprises KO_i and KI_i . It must also be noted that for odd rounds, the round data is passed through *FL*() and then *FO*(), while for even rounds it is passed through *FO*() and then *FL*().

2.1 Function FL

Function *FL*, as shown in Fig. 2, comprises a 32-bit data input I and a 32-bit subkey KL_i . The sub-key is split into two 16-bit parts, the upper $KL_{i,1}$ and lower $KL_{i,2}$. The input data I is split into two 16-bit strings as well, namely $L = I[31:16]$ and $R = I[15:0]$. We define:

$$R' = (R) \text{ XOR } [(L \text{ AND } KL_{i,1}) \lll 1]. \quad (4)$$

$$L' = (L) \text{ XOR } [(R' \text{ OR } KL_{i,2}) \lll 1]. \quad (5)$$

where AND : bitwise AND operation,
OR : bitwise OR operation,
<<<1 : left circular rotation by 1.

The 32-bit output of FL is provided from the concatenation of the 16-bit strings L' and R' .

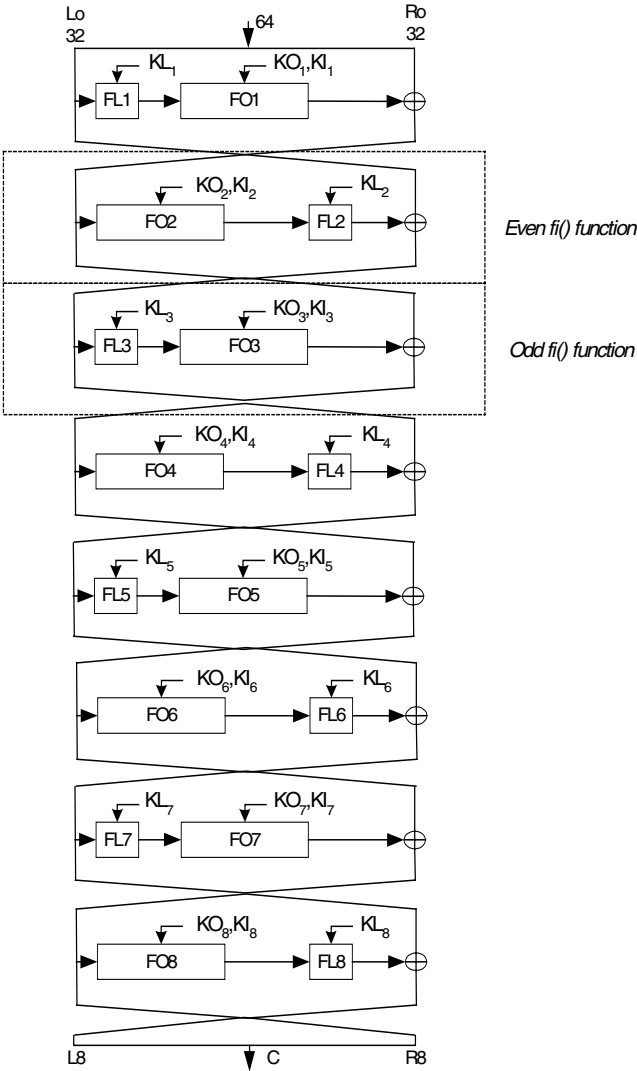


Fig. 1. The KASUMI algorithm

2.2 Function FO

Function FO , as illustrated in Fig. 2, comprises a 32-bit data input I and two sets of sub-keys, a 48-bit sub-key KO_i and 48-bit sub-key KI_i . As already mentioned, the 32-bit data input I is split into two 16-bit parts, the upper L_0 and the lower R_0 . The 48-bit subkeys are also divided into three 16-bit subkeys as follows:

$$KO_{i,1} = KO_i [47:32], KO_{i,2} = KO_i [31:16], KO_{i,3} = KO_i [15:0]. \quad (6)$$

and,

$$KI_{i,1} = KI_i [47:32], KI_{i,2} = KI_i [31:16], KI_{i,3} = KI_i [15:0]. \quad (7)$$

Then for each integer j with $1 \leq j \leq 3$ we define:

$$R_j = FI((L_{j-1}) XOR (KO_{i,j}, KI_{i,j})) XOR (R_{j-1}). \quad (8)$$

$$L_j = R_{j-1}. \quad (9)$$

The 32-bit output of FO is provided from the concatenation of the 16-bit strings L_3 and R_3 .

2.3 Function FI

Function FI , as shown in Fig. 2, takes a 16-bit data input I and a 16-bit sub-key $KI_{i,j}$. The input I is split into two unequal components, a 9-bit left half $L_0 = I [15:7]$ and a 7-bit right half $R_0 = I [6:0]$. Similarly the key $KI_{i,j}$ is split into a 7-bit component $KI_{i,j,1} = KI_{i,j} [15:7]$ and a 9-bit component $KI_{i,j,2} = KI_{i,j} [6:0]$. The function incorporates two S-boxes: S_7 that maps a 7-bit input to a 7-bit output, and S_9 that maps a 9-bit input to a 9-bit output. It also uses two additional operations which are named as $ZE(\cdot)$ and $TR(\cdot)$: $ZE(x)$ takes the 7-bit value x and converts it to a 9-bit value by adding two zero bits to the most-significant end. $TR(x)$ takes the 9-bit value x and converts it to a 7-bit value by discarding the two most-significant bits.

The following series of operations take place in the context of FI :

$$\begin{aligned} L_1 &= R_0, & R_1 &= (S_9[L_0] XOR ZE(R_0)) \\ L_2 &= (R_1) XOR (KI_{i,j,2}), & R_2 &= S_7[L_1] XOR TR(R_1) XOR KI_{i,j,1} \\ L_3 &= R_2, & R_3 &= S_9[L_2] XOR ZE(R_2) \\ L_4 &= (S_7[L_3]) XOR (TR(R_3)), & R_4 &= R_3 \end{aligned}$$

The function FI returns a 16-bit value delivered after the concatenation of the upper part L_4 and the lower part R_4 .

2.4 S-BOX

The S-boxes are non-linear substitution tables that have been designed so that they may be easily implemented in combinatorial logic as well as by a look-up table [2].

The input x comprises either seven (S7) or nine (S9) bits with a corresponding number of bits in the output y .

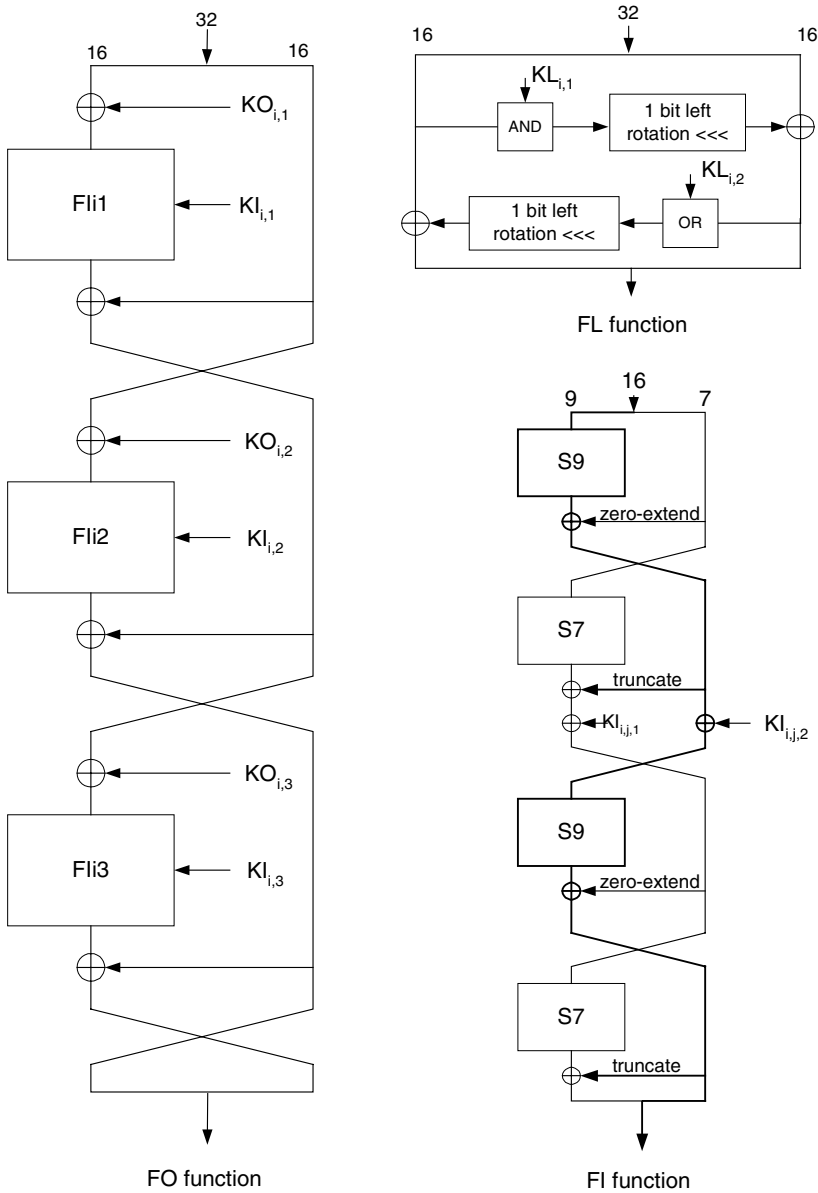


Fig. 2. KASUMI functions

3 Implementation of KASUMI on Various Hardware Platforms

Two different architectures have been investigated for the implementation of the KASUMI algorithm: a 2-round iterative, and an 8-round fully parallel architecture. These architectures are illustrated in Fig. 3.

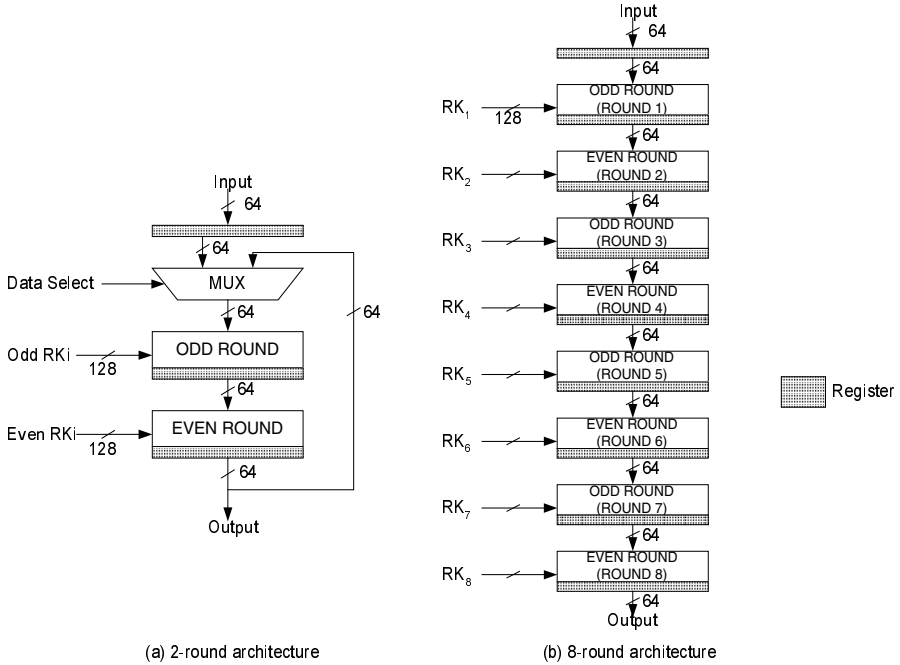


Fig. 3. Proposed architectures for KASUMI

As presented in section 2, the even rounds of KASUMI have a different structure than the odd rounds. The 2-round design implements one of each, i.e. one odd and one even round. The 8-round design implements the full KASUMI algorithm, by using four odd-round and four even-round instances. A trade-off between area and speed was made in each design.

The 2-round architecture in Fig. 3(a) was aimed at reducing the area required to implement the KASUMI algorithm. In this design, only one odd and one even round are used in an iterative scheme to implement KASUMI. This architecture requires only a quarter of the area of an 8-round fully unrolled KASUMI implementation, at the expense of speed and throughput. The output of each round is registered, and one additional register has been included at the input.

The second architecture, illustrated in Fig. 3(b) was aimed at maximizing speed and throughput, and implements the full 8 rounds of the KASUMI algorithm. This architecture uses internal pipelining by registering the outputs of each odd/even

round. Since there is no output feedback, a new 64-bit block can be loaded every clock cycle, hence an eight-fold increase in throughput is expected, compared to the area-optimized (2-round) scheme of Fig. 3(a). Of course, the cost of this significant performance gain is a four-fold increase in the utilization of hardware resources.

These two architectures have been implemented in three different hardware platforms, namely the Xilinx Virtex-E and Altera APEX 20KE Field Programmable Gate Arrays (FPGAs), as well as Atmel's 0.25 μ m ASIC Cell Based process (ATC25).

In order to take advantage of the architectural features of the FPGA technologies, we investigated the use of embedded memory to store the S7 and S9 SBOXes as look-up tables, apart from implementing them as combinational logic. The Xilinx Virtex-E FPGAs feature embedded blocks of memory called Block SelectRAM+, organized in columns at the left and right outside edges of the device. The Block SelectRAM+ are synchronous memory blocks, with registered outputs. The size of each block is 4,096 bits, and can be used in various configuration modes, from 4096x1-bit up to 256x16-bits. Various blocks can be merged to form wider or deeper memory structures as required. [5]

Altera follows a slightly different approach in their implementation of embedded memory. The APEX device family employs blocks of memory named Embedded System Blocks (ESBs), which span the entire device. The size of each ESB is 2,048 bits. The ESB can implement various types of memory blocks, including dual-port RAM, ROM, FIFO, and CAM blocks, with word sizes from 1-bit up to 16-bits. Several blocks can be merged in various configurations to implement fast and efficient FIFO buffers, and other memory structures. [6]

Both the Block SelectRAM+ (Xilinx) and the ESB (Altera) can implement logic functions when programmed with a read-only pattern during configuration, creating a large look-up table (LUT). With LUTs, combinatorial functions are implemented by looking up the results, rather than by computing them. This implementation of combinatorial functions can be faster than using algorithms implemented in general logic, a performance advantage that is further enhanced by the fast access times of the Block SelectRAM+ (Xilinx) and the ESB (Altera). The large capacity of these embedded memory blocks allows the implementation of complex functions in one logic level without the routing delays associated with linked logic elements (CLBs or LEs) or distributed RAM blocks.

The designs were implemented in behavioral VHDL. For the Xilinx technology, the Exemplar Leonardo Spectrum 2000.1a and Xilinx Foundation 3.1i tools were used for synthesis and implementation, respectively. For the Altera technology, the Altera Quartus 2000.05 tool was used for synthesis, implementation and timing analysis. For the Atmel 0.25 μ m ASIC implementation, the designs were synthesized using Synopsys Design Analyzer 3.11 [7]. All implementation runs were targeted at maximum speed with normal optimization effort, and constraints applied at the critical paths to reduce routing delays. We let the implementation tools place-and-route the designs automatically, without any manual floor planning or routing, so that we could also get an indicative overview of their standalone performance. It is expected that even higher performance can be achieved by manual floor planning and routing. All designs were fully tested, simulated and verified using the Implementors' Test Data [8].

The details regarding the actual implementations of the designs for the three target technologies are presented in the following sections, along with a discussion and analysis of the results.

3.1 Implementations for Xilinx Virtex-E

For the realization of the KASUMI algorithm in Xilinx FPGA technology, we selected the Virtex-E series as target devices. The Virtex-E FPGAs comprise of two major configurable elements: Configurable Logic Blocks (CLBs) and Input/Output blocks (IOBs). The CLBs are divided into two slices each and provide the functional elements for constructing logic. The IOBs provide the interface between the package pins and the CLBs. The CLBs interconnect through a general routing matrix (GRM, an array of routing switches located at the intersections of horizontal and vertical routing channels. The main reasons for the selection of Virtex-E series as target devices are that they incorporate fast and efficient routing resources (GRM), and a large number of internal memory blocks (Block SelectRAM+). This allowed us to implement both combinational and LUT-based SBOX versions of the KASUMI algorithm.

The *2_rounds_comb* version realizes the S7 and S9 SBOXes as pure combinational functions, while the *2_rounds_LUT* version realizes them as synchronous look-up tables, being mapped into *Block SelectRAM* internal memory components. The results for the Xilinx implementations of KASUMI are summarized in Table 1.

Table 1. Kasumi Implementation results for Xilinx Virtex-E

Architecture	Slices	FFs	Block RAMs	RAM bits	f_{\max} (MHz)	Device
<i>2_rounds_comb</i>	1287	280	n/a	n/a	20.88	XCV300E-6BG432
<i>2_rounds_LUT</i>	749	275	24	66048	35.35	XCV200E-6FG456
<i>8_rounds_comb</i>	4032	576	n/a	n/a	20.86	XCV400E-6BG432
<i>8_rounds_LUT</i>	2213	630	96	264192	37.72	XCV1000E-6BG560

[n/a = not applicable]

3.2 Implementations for Altera APEX 20KE

The Altera APEX 20KE Programmable Logic Devices (PLDs) combine the strengths of LUT-based and product-term-based devices with an enhanced memory structure. LUT-based logic provides optimized performance and efficiency for data-path, register-intensive, mathematical, or digital signal processing (DSP) designs. Product-term-based logic is optimized for complex combinatorial paths, such as complex state machines. Signal interconnections within APEX 20K devices (as well as to and from device pins) are provided by the FastTrack Interconnect—a series of fast, continuous row and column channels that run the entire length and width of the device.

The *2_rounds_comb* version realizes the S7 and S9 SBOXes as pure combinational functions, while the *2_rounds_LUT* version realizes them as synchronous look-up tables, being mapped into Embedded System Blocks (ESBs). The results for the Altera implementations of KASUMI are depicted in Table 2.

Table 2. KASUMI implementation results for Altera APEX 20KE

Architecture	LEs	FF	ESBs	RAM bits	f_{\max} (MHz)	Device
2_rounds_comb	2077	196	n/a	n/a	31.17	EP20K160EBC356-1X
2_rounds_LUT	821	263	48	66048	61.30	EP20K200EBC356-1X
8_rounds_comb	7106	576	n/a	n/a	26.24	EP20K200EBC356-1
8_rounds_LUT	2316	832	198	264192	40.5	EP20K1500EBC652-1X

[n/a = not applicable]

3.3 Implementations for ATMEL 0.25 μm

Apart from the implementations in FPGA technologies, synthesis results of the KASUMI algorithm were also obtained using *ATMEL's 0.25 μm ASIC Cell Based (ATC25) library* as a target. The Atmel ATC25 family is fabricated on a proprietary 0.25 micron five-layer-metal CMOS process intended for use with a supply voltage of $2.5\text{V} \pm 0.25\text{V}$. ATC25 is fully compatible with Atmel's extensive range of microcontrollers, DSPs, standard interfaces and Application Specific Cells, and support Memory Cells compiled to the precise requirements of the design for maximum area utilization.

Table 3 summarizes the results for the implementations of Kasumi in Atmel's 0.25u ASIC process. The results, presented in table, provide useful information for the evaluation of our implementation and the comparison of various technologies' performance metrics. The proprietary nature of the ATC25 library prevented us from having access to certain library cells (e.g. memory), so we were only able to implement the combinational-SBOX versions of KASUMI.

Table 3. KASUMI implementation results for Atmel 0.25u ASIC process

Architecture	Combinational Area (μm^2)	Non-Combinational Area (μm^2)	Total Area (*) (μm^2)	f_{\max} (MHz)
2_rounds_comb	14003.0	1678.0	15696.9	90.42
8_rounds_comb	43481.7	4136.5	47660.8	94.05

(*) Including the combinational, non-combinational and net interconnect area of the design.

3.4 Analysis of Results

As can be observed from Tables 1, 2 and 3 the total area for the 8 round implementations of KASUMI compared to the 2-round implementations, is greater by a factor of 3 to 3.5, rather than the anticipated factor of 4. This can be attributed to the capabilities of the synthesis tools to perform efficient resource sharing and other optimisations, leading to significant area reductions.

A significant increase in f_{\max} was expected in the LUT-based implementations, compared to the combinational versions. The Block SelectRAM+ in Xilinx and the ESB in Altera FPGAs are synchronous memories, with registered outputs. Effectively, these registers form additional pipeline stages and partition the critical path even further. As a result, the clock cycle is reduced, increasing the maximum frequency. These additional pipeline stages, however, increase the number of clock cycles required for the encryption/decryption of a 64-bit block. In the combinational

SBOX case, the number of clock cycles required for the encryption/decryption of a 64-bit block was $number_of_rounds+1$ (since there was an additional register at the input of the KASUMI module). With the incorporation of the synchronous memories for the SBOXes, the number of clock cycles effectively becomes $(5*number_of_rounds)+1$.

This has no impact in the 8-round fully unrolled KASUMI architecture; the initial latency increases from 9 clock cycles to 41, but once the pipeline fills we get a new encrypted/decrypted block at the output at every clock cycle. However, in the 2-round KASUMI architecture, a new block can be loaded only after the previous block has been fully processed. In this case, the incorporation of synchronous ROMs has a significant effect on the throughput.

Hence, for the 2-round KASUMI design, the effective throughput TP (Mbits/sec) is given by:

Combinational SBOX version:

$$TP = 64 / (\text{clock_period} * 8) = [64 \text{ (bits)} * f_{\text{max}} \text{ (MHz)}] / 8 . \quad (10)$$

Table 4. Throughput results for the implementations of KASUMI

Technology	Architecture	Clock Freq. f_{max} (MHz)	Cycles per block	Throughput (Mbits/s)
Xilinx	2_rounds_comb	20.88	8	167.04
	2_rounds_LUT	35.35	40	70.70
	8_rounds_comb	20.86	1	1335.04
	8_rounds_LUT	37.72	1	2414.08
Altera	2_rounds_comb	31.17	8	249.36
	2_rounds_LUT	61.30	40	122.60
	8_rounds_comb	26.24	1	1994.88
	8_rounds_LUT	40.50	1	3221.12
ASIC	2_rounds_comb	90.421	8	723.37
	8_rounds_comb	94.05	1	5786.94

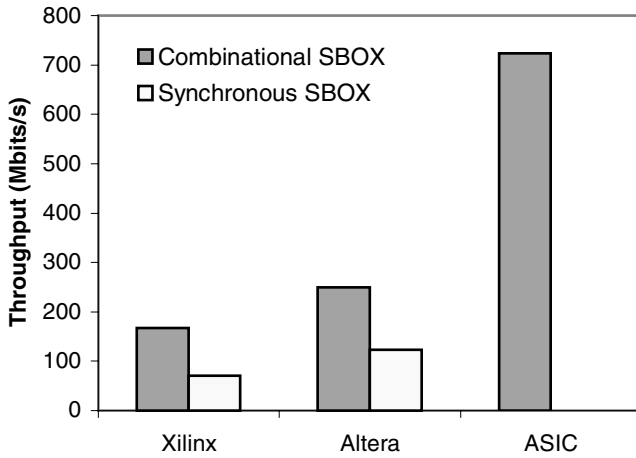
LUT-based SBOX version:

$$TP = 64 / (\text{clock_period} * 40) = [64 \text{ (bits)} * f_{\text{max}} \text{ (MHz)}] / 40 . \quad (11)$$

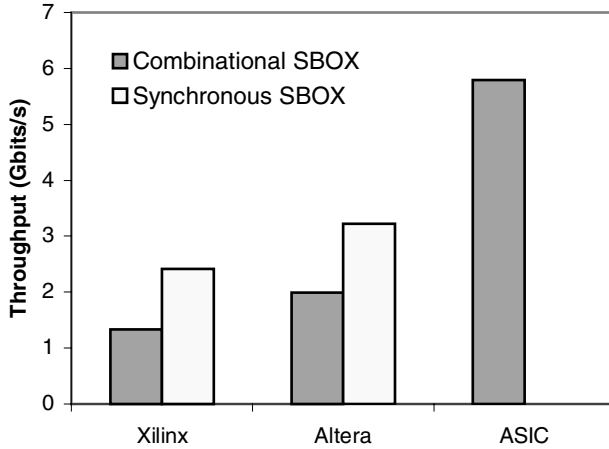
For the 8-round fully unrolled KASUMI design, the effective throughput TP (Mbits/sec), for both combinational and LUT-based SBOX implementations, is given by:

$$TP = 64 / \text{clock_period} = 64 \text{ (bits)} * f_{\text{max}} \text{ (MHz)} . \quad (12)$$

The throughput results for the various implementations of KASUMI on the Xilinx, Altera and Atmel hardware platforms are summarized in Table 4, and illustrated in Fig. 4.



(a) 2-round KASUMI implementation



(b) 8-round KASUMI implementations

Fig. 4. Throughput results from the implementations of KASUMI on various platforms

4 Design of the Confidentiality (f_8) and Integrity (f_9) Algorithms

Within the security architecture of the 3GPP system there are two standardized algorithms: A Confidentiality Algorithm f_8 , and an Integrity Algorithm f_9 . Both the f_8 and f_9 algorithms are based on the KASUMI algorithm (see section 2). The f_8 and f_9 are presented in the following sections.

4.1 Confidentiality Function f_8

The confidentiality algorithm f_8 is a stream cipher that is used to encrypt/decrypt blocks of data under a confidentiality key CK . The block of data may be between 1 and 5114 bits long. The algorithm uses KASUMI in a variant of the standard Output Feedback Mode (OFB) [9] as a keystream generator. The structure of the confidentiality function f_8 is illustrated in Fig. 5. [10]

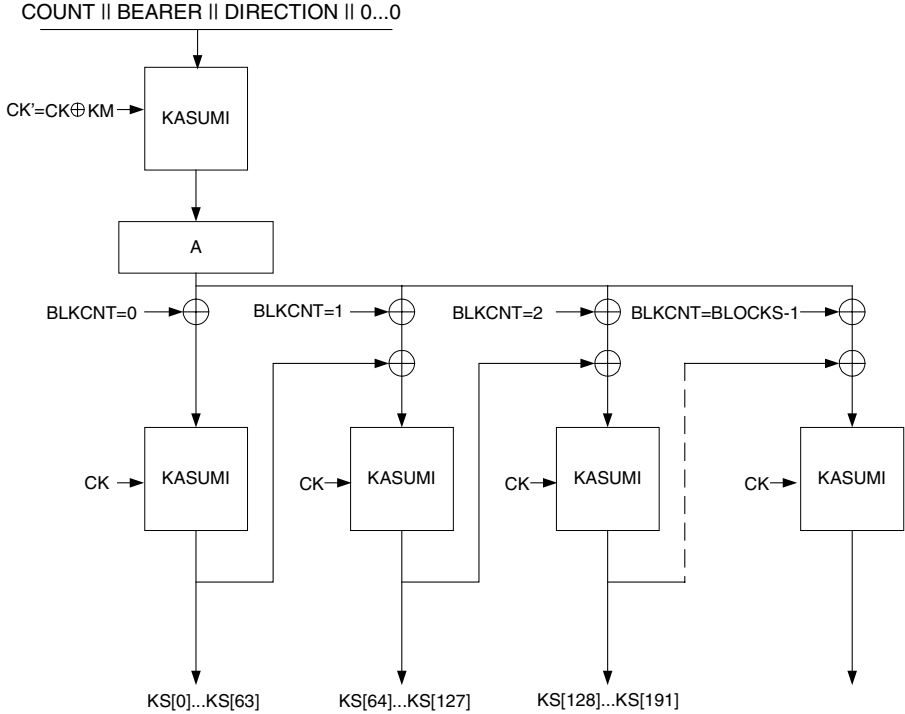


Fig. 5. Confidentiality Function f_8

During a pre-computation phase, the system parameters COUNT, BEARER and DIRECTION are padded to become a full length datablock and KASUMI encrypted with a derived key CK' . The derived key CK' is the exclusive-OR of the original cipher key CK and a fixed mask KM . The output of this process is a 64-bits register value A , which is part of the input in each subsequent KASUMI operation.

Subsequent 64-bit keystream blocks are then generated by running KASUMI in output feedback mode, with the register value A and the block counter (BLKCNT) as additional inputs to the feedback. The cipher text CT is then produced as the exclusive-OR of the keystream bits (KS) and the plaintext bits (PT), ie:

$$CT[i] = KS[i] \oplus PT[i], \quad \text{for } 0 \leq i \leq LENGTH - 1. \quad (13)$$

4.2 The Integrity Function f_9

The integrity function f_9 computes a 32-bit Message Authentication Code (MAC) on an input message under an integrity key IK . The message may be between 1 and 5114 bits in length. The structure of the f_9 function is depicted in Fig. 6.

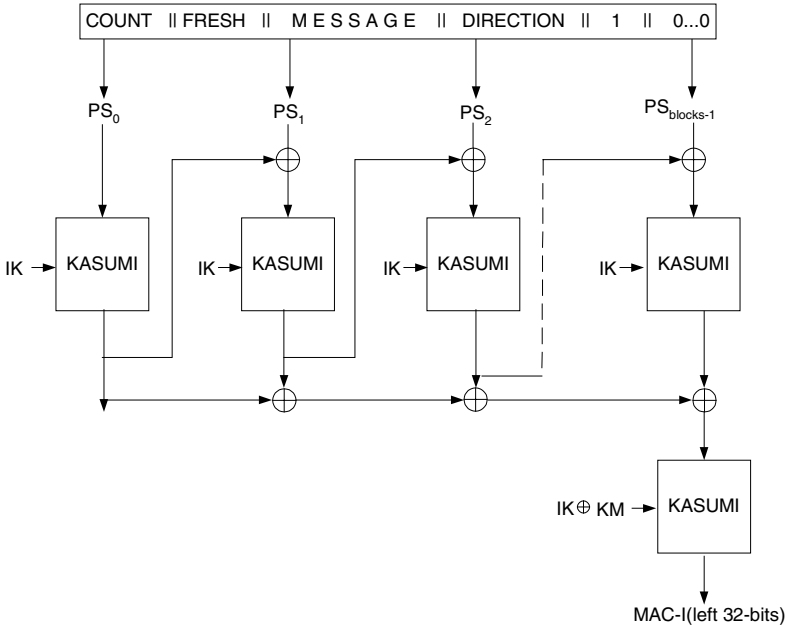


Fig. 6. Integrity Function f_9

The f_9 function is a variant of the standard CBC MAC as defined in ISO 9797 [11]. In this structure the exclusive-OR of the outputs from all block computations is exclusive-ORed to the input of the final KASUMI computation (output transformation). The MAC on the input message consists of the left half of the output from this last computation [10].

4.3 Implementation of the f_8 and f_9 Functions

As can be seen in Fig. 5 and Fig. 6, both the f_8 and f_9 functions use a number of cascaded KASUMI blocks in order to implement the Confidentiality cipher and the Message Authentication Code (MAC), respectively. Both algorithms must be able to handle messages up to 5114-bits long, in 64-bit blocks. Hence, implementing a full-scale architecture for these two algorithms would require up to 80 KASUMI modules. This would translate to a significant amount of area. In order to reduce this highly inefficient use of hardware resources, we propose two serial-like architectures for the implementation of the f_8 and f_9 functions, using only a small fraction of the area of a full-scale KASUMI implementation.

The proposed architectures for the f_8 and f_9 functions are illustrated in Fig. 7.

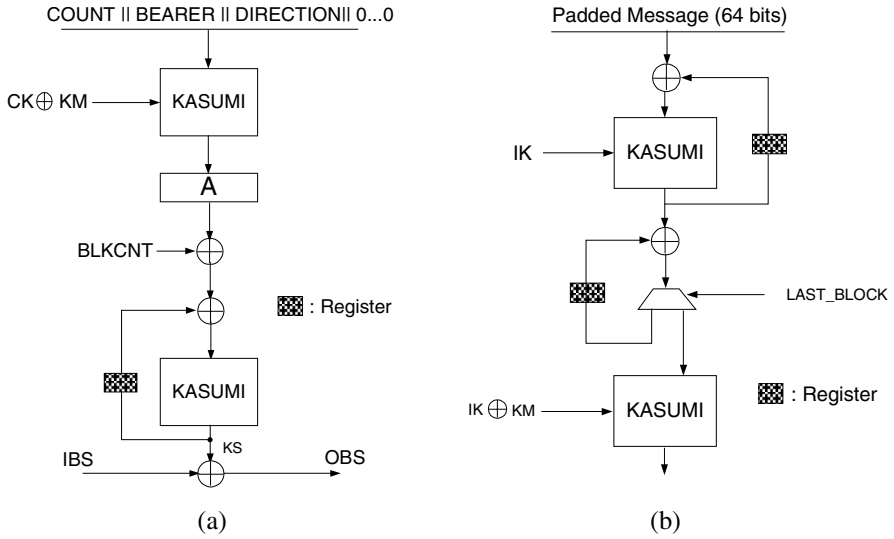


Fig. 7. Proposed designs for (a) the f_8 function, and (b) the f_9 function

As can be seen in Fig. 7 a single KASUMI block is used in an iterative scheme in order to implement the f_8 and f_9 functions. The number of blocks in the message (0 to 80) dictates the number of iterations that need to take place in each case. In the f_8 case (Fig. 7(a)), the datapath is basically controlled by means of a #BLOCKS counter. In the f_9 datapath (Fig. 7(b)) an additional multiplexer has been included, controlled by a LAST_BLOCK bit. This signal indicates whether the current iteration is the last one, hence forwarding the result to the last stage and then at the output.

We have experimented with various implementation options, regarding both the implementation of the KASUMI scheme (2- or 8-round schemes, see section 3), as well as the target technologies (Xilinx Virtex-E, Altera APEX 20KE, and Atmel 0.25 μ m). The results from the implementations for Xilinx and Altera FPGAs, and for Atmel’s ASIC process, are summarized in tables 5, 6 and 7, respectively. The 8-round KASUMI versions of f_8 and f_9 were implemented using combinational SBOXes only, since the amount of embedded memory that would be required to implement the SBOXes in LUTs exceeded the capacity of even the biggest parts in the target FPGA device families.

Table 5. f_8 and f_9 implementation results for Xilinx Virtex-E

Architecture	Slices	FFs	Block RAMs	f_{max} (MHz)	Device
$f_8_2_rounds_comb$	2781	800	n/a	20.52	XCV300E-6BG432
$f_8_2_rounds_LUT$	1563	784	48	33.14	XCV600E-6 BG432
$f_8_8_rounds_comb$	8146	1386	n/a	20.01	XCV1000E-6
$f_9_2_rounds_comb$	2671	731	n/a	20.68	XCV300E-6 BG432
$f_9_2_rounds_LUT$	1560	753	48	33.52	XCV600E-6 BG432
$f_9_8_rounds_comb$	8104	1323	n/a	20.19	XCV1000E-6BG560

[n/a = not applicable]

Table 6. f_8 and f_9 implementation results for Altera APEX 20KE

Architecture	LEs	FFs	ESB	f_{\max} (MHz)	Device
$f_8_2_rounds_comb$	4687	434	n/a	30.76	EP20K160EFC484-1
$f_8_2_rounds_LUT$	2128	568	100	49.50	EP20K400EBC652-1X
$f_8_8_rounds_comb$	15232	1386	n/a	29.06	EP20K400EBC652-1X
$f_9_2_rounds_comb$	4382	467	n/a	29.62	EP20K160EBC356-1
$f_9_2_rounds_LUT$	1901	601	100	51.79	EP20K400EBC652-1X
$f_9_8_rounds_comb$	13628	1323	n/a	28.93	EP20K400EBC652-1X

[n/a = not applicable]

Table 7. f_8 and f_9 implementation results for Atmel 0.25u ASIC process

Architecture	Combinational Area (μm^2)	Non-Combinational Area (μm^2)	Total Area (μm^2) (*)	f_{\max} (MHz)
$f_8_2_rounds_comb$	28918.75000	5376.50000	34328.30859	89.15
$f_8_8_rounds_comb$	89890.50000	12958.00000	102848.3156	90.09
$f_9_2_rounds_comb$	28082.75000	4453.00000	32567.69336	88.71
$f_9_8_rounds_comb$	86879.50000	9415.00000	96380.80469	89.12

(*) Including the combinational, non-combinational and net interconnect area of the design.

4.4 Discussion and Analysis of the Results

As was also discussed in section 3.4, an increase in f_{\max} was expected in both f_8 and f_9 implementations using LUT-based KASUMI blocks, because of the additional pipelining. The additional registers introduced by the incorporation of synchronous memories don't affect the throughput in the implementations that use 8-round KASUMI blocks. However, they do have a significant effect on the throughput in the 2-round KASUMI versions. The throughput of f_8 is a function of the block size, and is given by the following formula:

$$TP = (64 * \#Blocks) / [(\#Blocks + 1) * KASUMI_cycles * clock_period] = \quad (14)$$

$$= (64 * \#Blocks * f_{\max}) / [(\#Blocks + 1) * KASUMI_cycles] .$$

where, $KASUMI_cycles$ is the number of clock cycles required for the KASUMI encryption/decryption of a 64-bit block (see section 3.4). For the f_8 implementations using 2-round KASUMI blocks, formula (14) becomes:

Combinational SBOX version:

$$TP = (64 * \#Blocks * f_{\max}) / [(\#Blocks + 1) * 8] . \quad (15)$$

LUT-based SBOX version:

$$TP = (64 * \#Blocks * f_{\max}) / [(\#Blocks + 1) * 40] . \quad (16)$$

When using 8-round KASUMI blocks for the implementation of the f_8 function, the parameter $KASUMI_cycles$ in formula (14) effectively becomes 1. Hence, the throughput of f_8 using 8-round KASUMI block (with either combinational or LUT-based SBOXes) is given by:

$$TP = (64 * \#Blocks * f_{\max}) / (\#Blocks + 1) . \quad (17)$$

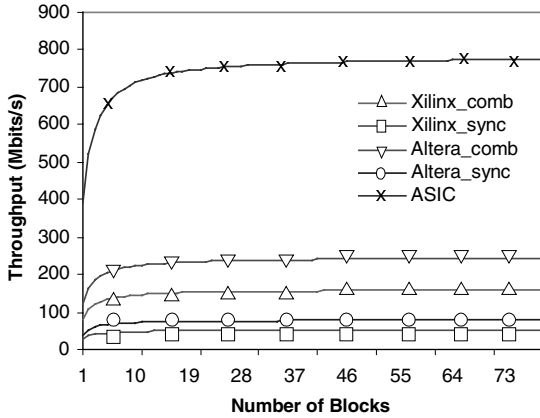
The throughput of f_8 for data block sizes from 1 to 5114 bits (1 to 80 64-bit blocks) on various hardware platforms is illustrated in Fig. 8.

The f_9 function calculates a Message Authentication Code (MAC) on an input message. As a performance metric for the implementations of the f_9 function we can compare the time required to compute a MAC on messages of varying sizes, from 1 to 5114 bits long (1 up to 80 64-bit blocks). The time required for the computation of a MAC is given by:

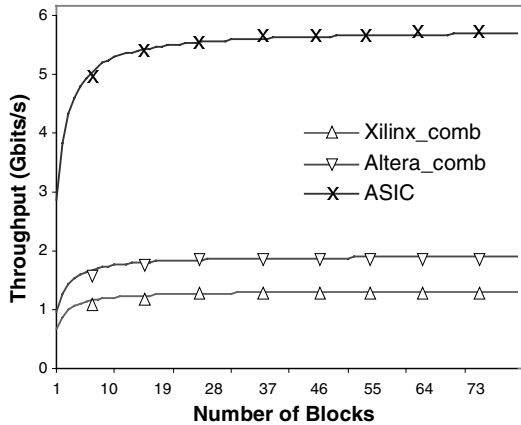
$$t_{\text{MAC}} = (\#Blocks + 1) * KASUMI_cycles * \text{clock_period} \quad (18)$$

$$= [(\#Blocks + 1) * KASUMI_cycles] / f_{\text{max}}.$$

where, $KASUMI_cycles$ is the number of clock cycles required for the KASUMI encryption/decryption of a 64-bit block (see section 3.4), and $\#Blocks$ is the number of 64-bit blocks in the message. The performances of the various implementations of the f_9 function are summarized in Fig. 9.



(a)



(b)

Fig. 8. Throughput of f_8 function using (a) 2-round KASUMI scheme, and (b) 8-round KASUMI scheme on various hardware platforms.

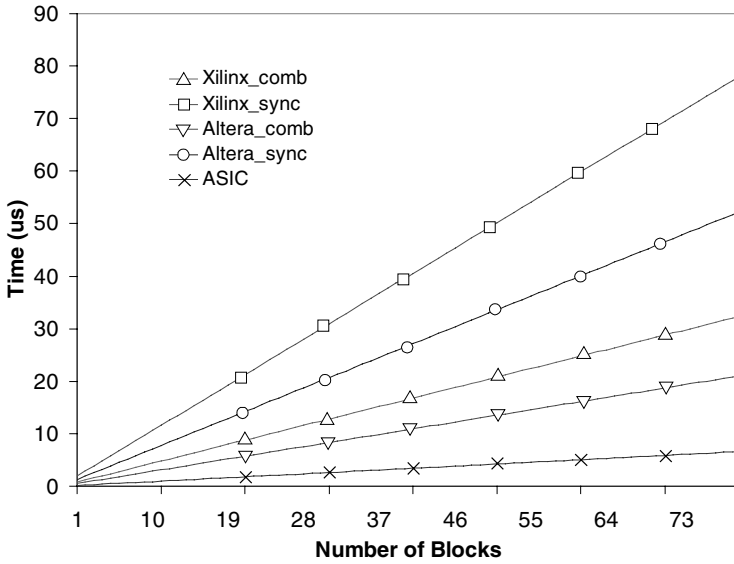


Fig. 9. Performance of f_9 function on various hardware platforms

5 Conclusions

In this paper, various hardware designs of the 3GPP Confidentiality and Integrity algorithms, as well as their core cryptographic engine KASUMI, have been presented. Design implementations have been performed for both FPGA (Xilinx, Altera) and ASIC (Atmel ATC25) platforms, exploring the performance of each architecture. Significant area reduction can be achieved by an iterative architecture scheme with a penalty on throughput. Furthermore, by substituting combinational functions – when possible - with LUTs, mapped in internal memory structures, higher operation frequency can be obtained.

References

1. ETSI/SAGE. f_8 and f_9 specification. Specification of the 3GPP Confidentiality and Integrity Algorithms Document 1, ETSI/SAGE, September 2000.
2. ETSI/SAGE. KASUMI specification. Specification of the 3GPP Confidentiality and Integrity Algorithms Document 2, ETSI/SAGE, December 1999.
3. 3G TS 33.102 V 3.1.0 3rd Generation Partnership Project. Technical Specification Group Services and System Aspects. 3G Security. Cryptographic Algorithm Requirements, December 1999.
4. Mitsuru Matsui: New block encryption algorithm MISTY. In *Fast Software Encryption '97*, vol. 1267 of LNCS, pages 54-68. Springer-Verlag, 1997.
5. Xilinx Data Book.

6. Altera Data Book.
7. Synopsys Design Analyzer Reference Manual v.2000.11.
8. ETSI/SAGE. Implementors' Test Data. Specification of the 3GPP Confidentiality and Integrity Algorithms Document 3, ETSI/SAGE, December 1999.
9. ISO/IEC 10116:1996: Information technology – Security techniques – Modes of operation for an n -bit block cipher algorithm.
10. ETSI/SAGE. 3GPP Standard Algorithms Task Force – Public Report – Security Algorithms Group of Experts (SAGE) Report on the Evaluation of 3GPP Standard Confidentiality and Integrity Algorithms.
11. ISO/IEC 9797-1:1999 (E). Information technology – Security techniques – Message Authentication Codes (MACs) – Part 1.

Efficient Implementation of Elliptic Curve Cryptosystems on an ARM7 with Hardware Accelerator

Sheng-Bo Xu and Lejla Batina

Pijnenburg SECUREALINK B.V.,
Boxtelseweg 26, 5261 NE,
Vught, The Netherlands
{s.xu, l.batina}@securealink.com

Abstract. This paper presents an efficient implementation of elliptic curve cryptosystems over a prime field on ARM7 with the help of a hardware accelerator. The hardware accelerator has two dedicated large number arithmetic units (LNAU's). Three different implementation platforms are considered: ARM7, ARM7 with one LNAU, and ARM7 with two LNAU's. The time costs for performing point multiplication are measured for all three platforms. On ARM7 with 2 LNAU's platform, we achieved the point multiplication in 18 ms with the chip running at 50 MHz clock frequency.

1 Introduction

Elliptic Curve Cryptography (ECC) was proposed independently by V. Miller [22] and N. Koblitz [14] in 1986. Since then, elliptic curve cryptography has been studied widely and thoroughly. A considerable amount of research has been done on its secure and efficient implementation. Recently, various studies of software and hardware implementations of elliptic curve cryptosystems have developed rapidly and their achievements have become a center of interest because of many advantages that ECC offers when compared with conventional public key cryptosystems. Namely, when comparing public-key cryptosystems, there are various relevant evaluation criteria such as: security, key lengths, speed, implementation issues, etc. Considering security, in recent work A. Lenstra and E. Verheul [17] showed that 1937-bit key size RSA may be considered to have an equivalent security as 190-bit key size ECC. In the same work, the authors estimated that 190-bit ECC keys are expected to be suitable for commercial security in the year 2021, so twenty years from now. Other benefits include: higher speed, lower power consumption and smaller certificates, which are especially useful within the application areas of the booming smartcard industry. Thus, it is expected that elliptic curve cryptosystems will be widely used for many security applications in the near future which necessarily increases a strong demand on efficient implementations.

Careful and extensive studies of ECC implementations in software and hardware on various platforms are beneficial to practitioners. For a software implementations of ECC over a field of characteristic 2 and also some large prime number p , many details are already known [30], [3], [9], [12], [18], [21], [29], [6]. A comparative study of software implementations on workstation of the NIST-recommended elliptic curves over prime fields was presented in [21] and over binary fields in [3]. Some hardware implementations were also presented already in 1993 such as [1]. However, as pointed out in [9], there is a lack of software implementations on restricted hardware resources.

Our contribution deals with such an implementation of ECC over a field of a prime characteristic. The implementation platform is ARM7 with a hardware accelerator, which is a modular-multiplier for RSA. Motivation for this work is in offering a migration path for RSA-users, which are still in the vast majority in present applications. Yet, previously mentioned advantages of elliptic curve cryptosystems, especially in constrained environments where it is not always feasible to use an RSA cryptosystem, make this study of interest.

We have implemented Elliptic Curve version of Diffie - Hellman scheme (ECDH) [10] and Elliptic Curve Digital Signature Algorithm (ECDSA) [10] on a 32-bit accelerator chip which provides fast acceleration for large number exponentiation, symmetric key and hash functions. The example of such platform is PCC-ISES [11]. Suitable for use in e-commerce server applications, the PCC-ISES features an embedded ARM7 RISC processor and embedded random number generator. Other applications include: secure e-transaction protocols, secure e-mail, virtual private network (VPN) servers etc. Maximal clock frequency is 50 MHz. Since PCC-ISES comprises two LNAU's to implement the Montgomery Modular Multiplication (MMM) for RSA, it has been chosen as an implementation platform for the modular multiplication operation in $GF(p)$ which is the most time consuming operation in elliptic curve arithmetic. We chose the NIST-recommended elliptic curve over a 192-bit prime field $p_{192} = 3D2^{192} - 2^{64} - 1$, i.e. p is a generalized Mersenne prime. This choice of prime allows an efficient modular reduction as described in [24]. This type of modular reduction turned out to be very efficient for a pure software implementation, but when implementing with the existing hardware we took advantage of Montgomery multiplication algorithm.

An elliptic curve over $GF(p)$ is specified by the coefficients $a, b \in GF(p)$ of its defining equation $y^2 = 3Dx^3 + ax + b$. The NIST curves all have $a = 3Dp - 3$ because this yields a faster algorithm for point doubling when using a special type of coordinates, so-called Jacobian coordinates. Details of the mathematical background of ECC are explained in the Section 4.

The remainder of this paper is organized as follows. Section 2 gives a survey of previous implementations of public-key algorithms on embedded microprocessors relevant for our work. In Section 3, we introduce the architecture of the targeted 32-bit implementation platform. Section 4 introduces the basic details of the underlying mathematics on elliptic curve cryptosystems. Section 5 describes the elliptic curve arithmetic and algorithms for the basic operations in a finite field.

In Section 6, the implementation details of EC arithmetic are given. Results and timings are given in Section 7. Security issues are considered in Section 8. Sections 9 and 10 conclude the paper.

2 Previous Work

This section reviews some of the most relevant previous contributions in implementations over the field of prime characteristic. In [7] is described an ECC implementation over a prime field on the 16-bit low-cost microcontrollers from a RISC based family. The authors have showed that it was possible to achieve acceptable performance even in highly constrained embedded systems. Namely, they obtained point multiplication in 3.4 seconds without any precomputation while the processor is running at 1 MHz.

In [9], a practical software ECDSA implementation over $GF(p)$ on the M16C, which is a 16-bit 10MHz microcomputer, is presented. The authors propose the use of special prime field of a characteristic $p = 3De2^c \pm 1$. This choice of field has some advantages for implementing multiplication in $GF(p)$ in a small amount of memory. They use a randomly generated curve with the coefficient $a = 3Dp - 3$. The speed of 150 m for generating and 630 m for verifying an ECDSA signature is reported by using sliding window method for fixed point multiplication.

Reference [12] proposes a new fast implementation method of public-key cryptography suitable for 200 MHz DSP. The first method is a modified implementation of the Montgomery multiplication, which is suitable for pipeline processing. For elliptic doubling, they devised a new method, which reduces the number of multiplications and additions in comparison with the one specified by [10]. They implemented both methods on the DSP TMS320C6201 (TI) which can operate eight function units in parallel and has a performance of 1600 MIPS at 200 MHz. The performance achieved was 11.7 ms for 1024-bit RSA signing and 3.97 ms for 160-bit ECDSA verification.

All above mentioned papers were considering implementations in the field $GF(p)$. Recently, two papers have introduced implementations on an 8-bit processors over Optimal Extension Fields (OEFs). Reference [5] reports on an ECC implementation over the field $GF(p^m)$, with $p = 3D2^{16} - 165$, $m = 3D10$. The authors achieve a performance of 122 ms for a 160-bit point multiplication on the CalmRISC running at 20MHz. For EC arithmetic they combine the mixed coordinates similar to those in [18]. The second paper [6] describes a smart card implementation over the composite field without the use of a coprocessor. Achieved performance is 1.95 s for a 134-bit fixed point multiplication and 8.37 s for a general point multiplication using the binary method of exponentiation.

When considering hardware implementations, work of [1] has already showed that ASIC's built for performing elliptic curve operations over the field $GF(2^{155})$ have only 12000 gates which occupy less than 5% of the common smart card chip real estate, which is difficult to achieve for RSA operations.

However, almost all work on this subject over prime fields is considering software implementations. It appears that the arithmetic in characteristic 2 is

easier to implement and area and power consumption are smaller than in the case of $GF(p)$. This is believed to be true, but only for platforms where specialized arithmetic coprocessors for finite field arithmetic are not available. In this work we hope to make a first step towards efficient, fast and secure implementation by taking advantage of an existing cryptographic accelerator.

3 32-Bit Platform: PCC-ISES

The PCC-ISES is an integrated circuit with an architecture that is very suitable for modular multiplication. This design contains two identical Large Number Arithmetic Units (LNAU), each designed as a systolic array. This array is one-dimensional and consists of a fixed number of Processing Elements.

The PCC-ISES has the following characteristics: Embedded Cryptographic Accelerators with 2 LNAU's capable of performing up to 2048-bit modular arithmetic, Embedded microprocessor ARM7, 128 KB embedded RAM, and other features required for various cryptographic applications.

It is well known that modular multiplication in a large prime field is the most time consuming operation in elliptic curve arithmetic. Montgomery's method is used in the modular exponentiator to implement the modular multiplication. Specific commands are defined for modular multiplication and exponentiation, which can be used by the ARM7 processor to access the modular exponentiator. In our implementation, we can use this dedicated hardware accelerator to implement the modular multiplication operation in $GF(p)$. Actually, this hardware accelerator can perform two modular multiplication operations at the same time, which provides the possibility of implementing elliptic curve arithmetic in parallel.

In addition, PCC-ISES has a true random number generator (TRNG) and SHA-1, which are necessary components for implementing ECDSA. Since TRNG, SHA-1, and the modular multiplication are implemented in separate hardware modules, the ARM7 processor is fully available for the embedded application software.

4 Elliptic Curves over $GF(p)$

In this work we are only considering non-supersingular elliptic curves over $GF(p)$ where p is a 192 bits long generalized Mersenne prime. In that case, an elliptic curve E is often expressed in terms of the Weierstrass equation:

$$y^2 = 3Dx^3 + ax + b \quad (1)$$

where $a, b \in GF(p)$ with $4a^3 + 27b^2 \neq 0 \pmod{p}$. This condition ensures that the discriminant:

$$D = 3D - 16(4a^3 + 27b^2) \quad (2)$$

is nonzero, or equivalent that the points $P = 3D(x, y)$ on the curve where $x, y \in GF(p)$ are nonsingular. The set of all these points (x, y) together with the point O (which is representing a neutral element of this group) and the operation of "addition", form an abelian group. (For details on elliptic curve "group operation" see for example [19], [15], [2].) There are many types of coordinates in which an elliptic curve may be represented. In the equation above affine coordinates are used, but so-called projective coordinates have some implementation advantages. The main conclusion is that point addition can be done in projective coordinates using only field multiplications, with no inversions required. Thus, inversions are deferred, and only one needs to be performed at the end of a point multiplication operation. A projective point (X, Y, Z) on the curve satisfies the homogeneous Weierstrass equation:

$$Y^2Z = 3DX^3 + aXZ^2 + bZ^3 \quad (3)$$

and, when $Z \neq 0$, it corresponds to the affine point $(X/Z, Y/Z)$. It appears that other projective representations result in more efficient implementations of the group operation. In particular, a weighted projective representation (also referred to as Jacobian representation) is preferred in the sense of faster arithmetic on elliptic curves ([2], [10]). In this representation a triplet (X, Y, Z) corresponds to the affine coordinates $(X/Z^2, Y/Z^3)$ for $Z \neq 0$. In this case we have a weighted projective curve equation of the form:

$$E : Y^2 = 3DX^3 + aXZ^4 + bZ^6. \quad (4)$$

4.1 Group Law

Thus, there is a well-defined group operation on the set of points on an elliptic curve defined over a finite field together with the point at infinity. These points form an abelian group with the addition operation. This addition is usually called the chord-and-tangent method and is defined as follows. In the case of the curve (1), the inverse of the point $P = 3D(x_1, y_1)$ is $-P = 3D(x_1, -y_1)$. Then the sum $P + Q$ of the points $P = 3D(x_1, y_1)$ and $Q = 3D(x_2, y_2)$ (assume that $P, Q \neq 0$, and $P \neq \pm Q$) is the point $R = 3D(x_3, y_3)$ where:

$$\begin{aligned} \lambda &= 3D \frac{y_2 - y_1}{x_2 - x_1} \\ x_3 &= 3D\lambda^2 - x_1 - x_2 \\ y_3 &= 3D(x_1 - x_3)\lambda - y_1. \end{aligned}$$

For $P = 3DQ$, we get the following, "doubling" formulae:

$$\begin{aligned}\lambda &= 3D \frac{3x_1^2 + a}{2y_1} \\ x_3 &= 3D\lambda^2 - 2x_1 \\ y_3 &= 3D(x_1 - x_3)\lambda - y_1.\end{aligned}$$

The point at infinity O plays a role analogous to that of the number 0 in ordinary addition. Thus, $P + O = 3DP$ and $P + (-P) = 3DO$ for all points P .

Weighted projective coordinates provide very fast arithmetic. Conversion from projective to affine coordinates costs 1 inversion and 4 multiplications, while vice versa is trivial. If one implements addition and doubling in a way specified in [10], the total costs for general addition is $1I + 3M$ in affine coordinates and $16M$ in projective coordinates ($11M$ if $Z_1 = 3D1$ i.e. one point is given in affine coordinates, and the other one in projective coordinates). In the case of doubling (with $a = 3Dp - 3$), this relation is $1I + 4M$ in affine coordinates against $8M$ in projective coordinates. (Here, I and M are standing for the modular inversion and multiplication operations, respectively). Thus, the choice of coordinates is determined by the ratio $I : M$. Only if $I / M < 4$, which is very unlikely, affine coordinates might be a better option than other coordinates. Hence, the weighted projective coordinates turned out to be the best choice on our platform.

When working with Jacobian coordinates there is a possibility to improve the performance by exploiting the parallel nature of computation as explained in [26]. This parallel execution may be limited due to SIMD based constraints which does not apply to our platform and one can obtain roughly a 50-60 percent improvement in performance. In the same paper the author has shown that using the Hessian form (see also [4], [13]) of an elliptic curve allows one to implement the point addition and point doubling operation in a highly parallel way. This can be especially exploited by a processor architecture with multiple arithmetic logic units (ALU's), such as the PCC-ISES.

4.2 The Hessian Form of an Elliptic Curve

A Hessian elliptic curve over some field K is a plane cubic curve given by the following equation in projective coordinates:

$$X^3 + Y^3 + Z^3 - 3mXYZ = 3D0, \quad (5)$$

Where $m \in K$ and $m^3 \neq 1$. This condition $m^3 \neq 1$ is necessary and sufficient for the curve to be nonsingular and hence elliptic. It can be transformed to the canonical form (1) by change of variables as explained in [26].

The reverse process i.e. transformation of the canonical form to the Hessian form is not always possible. To make construction possible, to adapt canonical form of equation (1) to a Hessian form (5), some restrictions on the field and

curve are recommended. A finite field $GF(p)$ must satisfy that $p \equiv 2 \pmod{3}$ so that every element in the field has a unique cube root, to make use of special formulae by which the curve (1) is transferred to the Hessian form. Also, a rational point of order 3 on the curve is required for a change of variables which results in the Hessian form. A complete description is given in [26].

The group law on curves in Hessian form has been known for a long time [4]. The group operation with formulae as given in [26] would allow us to implement the group operation in a highly parallel way by use of up to three LNAU's.

5 EC Arithmetic and Modular Arithmetic in $GF(p)$

Due to the space limitation, some implementation algorithms for EC arithmetic and modular arithmetic in $GF(p)$ are not listed in this paper. For more details on the algorithms the reader is referred to literature.

5.1 EC Arithmetic

Section 4 has described the basic mathematics of EC arithmetic, point addition and point doubling. Since projective coordinates provide fast implementation on the PCC-ISES, Algorithms A.10.4 and A.10.5 specified by IEEE [10] are employed to perform point doubling and addition, respectively. In addition, parallel algorithms given in [26] are employed to perform point addition and doubling on an ARM7 with two LNAU's platform.

Scalar multiplication (also called *point multiplication*) can be performed by repeatedly using point doubling and/or addition operations. It is well known that NAF representation reduces the number of point addition operations on account of subtraction in scalar multiplication. In our implementation, Algorithm A.10.9. specified by IEEE [10] is employed to perform scalar multiplication, in which a simple balance coding method is applied.

5.2 Modular Arithmetic in $GF(p)$

In order to perform EC arithmetic, we need modular arithmetic in $GF(p)$, which includes modular addition, subtraction, multiplication, squaring, and inversion.

Modular addition and subtraction operations are simple in small size fields. In a large size field, for example p is 192-bit long, multi-precision algorithms ([3], [20]) are needed to perform modular addition and subtraction operations.

There are different methods for performing modular reduction. In the case when p is a generalized Mersenne prime, fast reduction can be achieved by using Algorithm given in [21]. Algorithm 7 [21] is used in our ARM software solution while on the PCC-ISES it is implemented by means of Montgomery's multiplication (by 1).

Modular multiplication can be also implemented by different methods. One method, which is particularly attractive for software implementation when the modulo p is a generalized Mersenne prime, can be found in [3].

For a hardware implementation of modular multiplication, Montgomery's Multiplication Method (MMM) [23] is more efficient than previously mentioned method. For a word base $b = 3D2^\alpha$, R has to be chosen such that $R = 3D2^r = 3D(2^\alpha)^t > N$.

Montgomery's method for multiplying two integers x and y modulo N , avoids trial division by N which is the most expensive operation in hardware. When computing the Montgomery product $T = 3DxyR^{-1} \bmod N$, the following procedure is performed ([20]):

Algorithm 1. Montgomery modular multiplication

INPUT: Integers $N(\text{odd})$, $x \in [0, N - 1]$, $y \in [0, N - 1]$, $R = 3D = 2^r$, and $N' = 3D - N^{-1} \bmod 2^\alpha$
 OUTPUT: $xyR^{-1} \bmod N$
 1. $T \leftarrow 0$.
 2. For i from 0 to $(t-1)$ do:
 2.1 $m_i \leftarrow (t_0 + x_i y_0) N' \bmod 2^\alpha$
 2.2 $T \leftarrow (T + x_i y + m_i N) / 2^\alpha$
 3. If $T \geq N$, then $T \leftarrow T - N$
 4. Return (T)

In the original proposal of Montgomery after each multiplication a reduction was needed (step 3 in the algorithm above). The input had the restriction $X, Y < N$ and the output T was bounded by $T < 2N$. The result of this is that in the case $T > N$, N must be subtracted so that the output can be used as input of the next multiplication. To avoid this subtraction a bound for R is known ([28]) such that for inputs $X, Y < 2N$ also the output is bounded by $T < 2N$. In practice this means that the output of the multiplication can be directly used as an input of the next Montgomery multiplication. Hence, when $X, Y < 2N$ and $R > 4N$, The result of a Montgomery multiplication $XYR^{-1} \bmod N < 2N$. So, Algorithm 1 is implemented in the PCC-ISES, without Step 3, which is required only at the end of exponentiation. This algorithm is used in the PCC-ISES for an RSA exponentiation, in which case N is up to 2048-bit large composite number. For an ECC operations N is replaced by a 192-bit prime p .

PCC-ISES has dedicated hardware - Large Number Arithmetic Unit (LNAU) - to do modular multiplication with MMM. One modular multiplication requires two MMM operations.

Since modular inversion is working in a prime field $GF(p)$ in this case, calculating the multiplicative inverse of a non-zero element x can be carried out as follows, $x^{-1} \equiv x^{p-2} \bmod p$, i.e. by use of Fermat's Little theorem. This can be easily achieved on PCC-ISES using the Montgomery method. For software implementation, a variant of the Binary Extended Euclidean algorithm given in [21] can be used to calculate the multiplicative inverse of every non-zero element in $GF(p)$.

6 Implementations

6.1 EC Domain Parameters

The underlying field $GF(p)$ is usually fixed for all entities, and each entity selects its own elliptic curve E and point $G \in E(GF(p))$. In this case, the defining equation for E , the point G , and the order r of G must also be included in the entity's public key. If the underlying field is fixed, then hardware and software can be built to optimize computations in that field. We used the field $GF(p)$ ([24]), where $p = 3D2^{192} - 2^{64} - 1 \equiv 2(mod 3)$.

6.2 Three Different Implementations of EC Arithmetic

The elliptic curve arithmetic requires the basic modular arithmetic operations in $GF(p)$. Since modular multiplication and inversion in a large prime field are the most time consuming operations and modular inversion is only performed once, the implementation of modular multiplication is the main criterion when choosing a platform. On PCC-ISES, modular multiplication and inversion can be performed on two different platforms: one is the ARM7 processor only and the other is with the hardware accelerator (LNAU).

On ARM-only platform, modular multiplication is performed by means of integer multiplication and reduction algorithms [21]. ARM7 processor has an efficient multiplier, which can calculate the 64-bit product of two 32-bit numbers. In addition, modular inversion operation is performed using the Binary Extended Euclidean algorithm.

On ARM7 with one LNAU platform, modular multiplication and inversion = are performed on the LNAU. Specific commands are defined for modular = multiplication and exponentiation, which can be used by ARM7 processor to access the modular exponentiator. The other operations are performed on ARM7 = processor.

On ARM7 with two LNAU's platform, two LNAU's are able to perform two separate modular multiplications at the same time. Thus, we can implement point addition and doubling in a parallel way by use of algorithms similar to those given in [26].

The other modular operations are performed on ARM7. ARM software is written in C code.

7 Results and Timings

7.1 Time Costs of Multiplication and Inversion in $GF(p)$

The modular multiplication and exponentiation functions on the PCC-ISES are used to execute multiplication and inversion. The time costs of multiplication and inversion in $GF(p)$ are listed in Table 1.

Table 1. Time costs for performing modular multiplication = and inversion on the LNAU.

Operations in $GF(p)$	Time cost
Modular multiplication	$5\ \mu s$
Modular exponentiation	$291\ \mu s$

7.2 Time Costs of Scalar Multiplication

We implemented the scalar multiplication in three different platforms: ARM7, ARM7 with one LNAU, and ARM7 with two LNAU. The time costs are listed in Table 2, respectively.

Table 2. Time costs of scalar multiplication in three = different platforms.

Platforms	Time cost
ARM7	300 ms
ARM7+ 1 LNAU	30 ms
ARM7+ 2 LNAU	18 ms

7.3 Time Costs of ECDH and ECDSA Protocols

Since the hardware accelerator has an embedded random number generator and SHA-1 core, it is very easy to implement ECDH and ECDSA on this implementation platform.

The time costs of ECDH and ECDSA on the three implementation platforms are measured and listed in Table 3.

Table 3. Time Costs of ECDH and ECDSA.

Platforms	ECDH	ECDSA sign	ECDSA verify
ARM7	300 ms	305 ms	612 ms
ARM7+1 LNAU	30 ms	32 ms	69 ms
ARM7+2 LNAU	18 ms	19.5 ms	39.5 ms

8 Security Remarks

When considering efficient implementation one has to focus on the side channel attacks, first of all timing ([8]) and power analysis attacks ([16]).

Namely, computations performed in non-constant time i.e. computations which are time-dependent on the values of the operands, may leak secret key information. This observation is the basis for timing attacks. On the other hand,

power analysis based attacks use the fact that the power consumed at any particular time during a cryptographic operation is related to the function being performed and (possibly sensitive) data being processed.

However, as a result of several contributions, some recommendations have been given for a public key cryptography, such as using the method of Montgomery for multiplication. The reason for that is following: a modular reduction step may also be vulnerable to timing attack and in the case of this method at most one modular reduction is introduced for every multiplication or per step of exponentiation. In our implementation with cryptographic accelerator even these reductions are excluded. The weaknesses in the conditional statements of the algorithm (used for realization of the reduction step) are time variations and therefore these should be omitted. By use of an optimal upper bound the number of iterations required in the algorithm based on Montgomery's method of multiplication can be reduced ([28]). More precisely, some savings in hardware that have been included in presented architecture avoid the conditional statements while performing the exponentiation. In that way, the implementation of modular exponentiation operates in constant time, which is presumed to significantly reduce the potential risk of the timing attack.

When considering power analysis attacks, some precautions have also been introduced. The exponent re-coding method ([20]), is more difficult to attack than the standard square-and-multiply method for exponentiation and the fact that certain number of LNAU's operate in parallel (including their Processing Elements) make these types of attacks far less likely to succeed.

Hence, the use of existing hardware has more advantages when compared with software-only implementation in the ARM7. However, to address security issues properly is beyond the scope of this paper and will be treated elsewhere in detail.

9 Future Work

When EC domain parameters are fixed in some applications, such as, the implementation of WAP in mobile terminals, the real-time calculation of scalar multiplication can be speeded up by means of the sliding window method [2] with pre-computed look-up table. The expected running time of this algorithm is approximately $(D + ((2^{w-2} - 1)A) + (m/(w+1)A + mD))$ according to [21]. Here D and A represent the time costs of point doubling and addition, respectively and m is the size of the prime field and w is the width of sliding window. Using this technique, it is expected that the implementation of elliptic curve cryptosystems can be speeded up four times at the cost of some pre-computation and extra ROM storage.

10 Conclusions

We have described three practical implementations of an EC cryptosystem over a prime field, where the prime is a generalized Mersenne prime, on the ARM7.

We have started with a pure software implementation, which is afterwards improved by using the cryptographic accelerator for modular arithmetic. The PCC-ISES has the best performance of 1024-bit RSA processing (compared to other chips of similar characteristics) as mentioned also by the author of [25], which made this platform suitable for this type of implementation. We compared the results of implementing ECC on one LNAU with taking advantage of some parallelism in a way of full use of both LNAU's. The achieved results for one LNAU can be improved more than 40% when using both LNAU's. This platform appeared to be not just more efficient, but also more secure in the sense of timing and power analysis attacks. It is expected that a new design with three LNAU's available will give further improvements in performance while maintaining the favorable security characteristics.

Acknowledgements. The authors would like to thank Cees Jansen and Geeke Muurling for useful discussions and other members of Crypto Research group in Securealink for the stimulating atmosphere. We are especially grateful to Kees de Laat for help with the implementations and timings. We also thank the anonymous referees for helpful comments.

References

1. G.B. Abnew, R. C. Mullin, and S. A. Vanstone. An implementation of elliptic curve cryptosystems over $GF(2^{155})$. *IEEE Journal on SAC*, 11(5), June 1993.
2. I.Blake, G. Seroussi, and N. Smart. *Elliptic Curves in Cryptography*. Cambridge University Press, 2000.
3. M.Brown, D. Hankerson, J.Hernandez, and A. Menezes. Software implementation of elliptic curve cryptography over binary fields. *Lecture Notes in Computer Science, Springer-Verlag*, 2000. Cryptographic Hardware and Embedded Systems - CHES 2000.
4. J.W.S. Cassels. *Lectures on Elliptic Curves*. LSM Student Texts, Cambridge University Press, 1991.
5. J.W. Chung, S. G. Sim, and P. J. Lee. Fast implementation of elliptic curve defined over $GF(p^m)$ on C alm RISC with MAC 2424 coprocessor. In C. K. Koc. and C. Paar, editors, *Proceedings of the Second Workshop on Cryptographic Hardware and Embedded Systems - CHES '00*, pages 57–70, 2000.
6. D. V. Bailey, D. Woodbury and C. Paar. Elliptic curve cryptography on smart cards without coprocessors. In *IFIP CARDIS 2000, Fourth Smart card Research and Advanced Application Conference*, Bristol, UK, September 20-22 2000. Kluwer, 2000.
7. J. Guajardo, R. Bl. umel, U. Krieger, and C. Paar. Efficient implementation of elliptic curve cryptosystems on the TI MSP 430x33x family of microcontrollers. *Fourth International Workshop on Practice and Theory in Public Key Cryptography - PKC 2001*, 2001.
8. G. Hachez, F. Koeune, and J.-J. Quisquater. Timing attack: what can be achieved by a powerful adversary? *Proceedings of the 20th symposium on Information Theory in the Benelux*, pages 63–70, May 1999.

9. T. Hasegawa, J. Nakajima, and M. Matsui. A practical implementation of elliptic curve cryptosystems over $GF(p)$ on a 16-bit microcomputer. In *Hideki Imai and Yuliang Zheng, editors, First International Workshop on Practice and Theory in Public Key Cryptography - PKC '98*, LNCS 1431:182–194, 1998. Springer-Verlag.
10. IEEE. IEEE P1363 standard specifications for public key cryptography. 1999.
11. ISES. PCC-ISES datasheet, [www.secure-a-link.com /pdfs/isespdf/isesdata.pdf](http://www.secure-a-link.com/pdfs/isespdf/isesdata.pdf).
12. K. Itoh, M. Takenaka, N. Torii, S. Temma, and Y. Kurihara. Fast implementation of public-key cryptography over $GF(p)$ on a 16-bit microcomputer. In *C. K. Koc and C. Paar, editors, Proceedings of the First Workshop on Cryptographic Hardware and Embedded Systems - CHES '99*, LNCS 1717:61–72, 1999. Springer-Verlag.
13. M. Joye and J. J. Quisquater. Hessian elliptic curves and side-channel attacks. In *C. K. Koc and D. Naccache and C. Paar, editors, Proceedings of the Third Workshop on Cryptographic Hardware and Embedded Systems - CHES '01*, 2001.
14. N. Koblitz. Elliptic curve cryptosystem. *Mathematics in Computation*, (48):203–209, 1987.
15. N. Koblitz, A. Menezes, and S. Vanstone. The state of elliptic curve cryptography. *Designs, Codes and Cryptography*, (19):173–193, 2000.
16. P. Kocher, J. Jaffe, and B. Jun. Differential power analysis. *Lecture Notes in Computer Science*, Springer-Verlag, pages 388–397, 1999. Advances in Cryptology-CRYPTO 99.
17. A. Lenstra and E. Verheul. Selecting cryptographic key sizes. In *Hideki Imai and Yuliang Zheng, editors, Third International Workshop on Practice and Theory in Public Key Cryptography - PKC 2000*, LNCS 1751.
18. C. H. Lim and H. S. Hwang. Fast implementation of elliptic curve arithmetic in $GF(p^n)$. In *Hideki Imai and Yuliang Zheng, editors, Third International Workshop on Practice and Theory in Public Key Cryptography - PKC 2000*, LNCS 1751:405–421.
19. A. Menezes. *Elliptic curve public key cryptosystems*. Kluwer Academic Publishers, 1993.
20. A. Menezes, P. van Oorschot, and S. Vanstone. *Handbook of applied cryptography*. CRC Press, 1997.
21. A. J. Menezes, D. Hankerson, J. Lopez, and M. Brown. Software implementation of the NIST elliptic curves over prime fields. In *D. Naccache, editors, Topics in Cryptology - CT - RSA 2001, The Cryptographer's Track at RSA Conference 2001 San Francisco, CA, USA, April 8-12, 2001*, LNCS 2020, 2001.
22. V. Miller. Use of elliptic curves in cryptography. *Advances in Cryptography - Crypto '85*, LNCS 218:417–426, 1986. Springer-Verlag.
23. P. Montgomery. Modular multiplication without trial division. *Mathematics of Computation*, Vol. 44:519–521, 1985.
24. NIST. NIST Documents, recommended elliptic curves for federal government use. July 1999.
25. H. Nozaki, M. Motoyama, A. Shimbo, and S. Kawamura. Implementation of RSA algorithm based on RNS montgomery multiplication. In *C. K. Koc and D. Naccache and C. Paar, editors, Proceedings of the Third Workshop on Cryptographic Hardware and Embedded Systems - CHES '01*, 2001.
26. N. P. Smart. The Hessian form of an elliptic curve. In *C. K. Koc and D. Naccache and C. Paar, editors, Proceedings of the Third Workshop on Cryptographic Hardware and Embedded Systems - CHES '01*, 2001.
27. C.D. Walter. Safely reducing the number of iterations in Montgomery modular multiplication. 2001. preprint.

28. C.D. Walter and S. Thompson. Distinguishing exponent digits by observing modular subtractions. *Lecture Notes in Computer Science*, Springer-Verlag, (2020):192–207, 2001. Topics in Cryptology - CT-RSA 2001.
29. E. De Win, A. Bosselaers, and S. Vandenberghe. A fast software implementation for arithmetic operations in $\text{GF}(2^n)$. *Advances in Cryptology - Proceedings of Asiacrypt '96, Lecture notes in computer science*, LNCS 1163:65–76, 1996.
30. E. De Win, S. Minster, and M. Wiener. On the performance of signature schemes based on elliptic curves. *Algorithmic Number Theory, Proceedings Third Intern. Symp.*, ANTS-III(1423):252–266, 1998.

A Theoretical DPA-Based Cryptanalysis of the NESSIE Candidates FLASH and SFLASH

Rainer Steinwandt, Willi Geiselmann, and Thomas Beth

IAKS/E.I.S.S., Fakultät für Informatik, Universität Karlsruhe,
Am Fasanengarten 5, 76 131 Karlsruhe, Germany

Abstract. Within the NESSIE (New European Schemes for Signatures, Integrity, and Encryption) project, the signature schemes FLASH and SFLASH have been proposed for the use on low-cost smartcards.

We show theoretically how differential power analysis (DPA) can be used to reveal the complete secret key in possible smartcard implementations of FLASH and SFLASH. To our knowledge no smartcard implementations of these schemes are available at the moment, so an experimental verification of this attack has not been done so far.

1 Introduction

In [4,5,6] two asymmetric signature schemes for the use on low-cost smartcards, called FLASH resp. SFLASH, have been proposed. Both schemes have been accepted as candidates within the NESSIE (New European Schemes for Signatures, Integrity, and Encryption) project.

Subsequently we demonstrate that the design of FLASH and SFLASH is highly susceptible to an attack making use of differential power analysis (DPA) (cf. [2] for an explanation of this technique). Our attack does not aim at the mathematical primitives of FLASH and SFLASH; instead, we show that an implementation of these algorithms on smartcards is very likely to be insecure. Namely, based on certain assumptions on the underlying implementation, we describe how DPA can be applied to reveal the complete secret key in the aforementioned signature schemes. We are not aware of existing smartcard implementations of these schemes, so no experimental verification of our approach has been done so far.

Nevertheless, as for cryptographic smartcard applications robustness against this kind of attack is of crucial importance, and we do not know any suggestions to fix these problems in FLASH resp. SFLASH, we dissuade from using these NESSIE candidates in the proposed form.

2 Description of SFLASH

Both FLASH and SFLASH make use of multivariate polynomials over finite fields, and they can be seen as so-called C^{*-} algorithms with a special choice of the parameters (cf. [3]). As they are very similar in design, in the sequel we

focus on SFLASH where our attack is slightly simpler to explain; in Section 4 we shortly discuss the (straightforward) adaptations when dealing with FLASH instead of SFLASH. Moreover, we restrict our description to those aspects of SFLASH which are relevant for explaining our attack, for a more detailed description we refer to the SFLASH specification [5].

2.1 Parameters of the Algorithm

SFLASH makes use of three finite fields along with corresponding bijections:

- $K := \mathbb{F}_2[X]/(X^7 + X + 1)$ (which is isomorphic to \mathbb{F}_{128}) along with the bijection

$$\begin{aligned} \pi : \quad \{0, 1\}^7 &\longrightarrow K \\ (b_0, \dots, b_6) &\longmapsto \sum_{i=0}^6 b_i X^i \pmod{X^7 + X + 1} \end{aligned}$$

- $K' := \pi(\{0, 1\} \times \{0\}^6)$ (which is isomorphic to \mathbb{F}_2)
- $L := K[X]/(X^{37} + X^{12} + X^{10} + X^2 + 1)$ along with the bijection

$$\begin{aligned} \varphi : \quad K^{37} &\longrightarrow L \\ (b_0, \dots, b_{36}) &\longmapsto \sum_{i=0}^{36} b_i X^i \pmod{X^{37} + X^{12} + X^{10} + X^2 + 1} \end{aligned}$$

Secret key. The secret key consists of three parts:

- $\Delta \in \{0, 1\}^{80}$: a secret 80-bit string
- $s = (S_L, S_C)$: an affine bijection $K^{37} \longrightarrow K^{37}$ given by a 37×37 matrix $S_L \in K'^{37 \times 37}$ and a column vector $S_C \in K'^{37}$
- $t = (T_L, T_C)$: an affine bijection $K^{37} \longrightarrow K^{37}$ given by a 37×37 matrix $T_L \in K'^{37 \times 37}$ and a column vector $T_C \in K'^{37}$

For deriving the corresponding public key we also need the function

$$\begin{aligned} F : L &\longrightarrow L \\ \alpha &\longmapsto \alpha^{128^{11} + 1}. \end{aligned}$$

Moreover, for a bitstring $\lambda = (\lambda_0, \dots, \lambda_m)$ and integers $0 \leq r \leq s \leq m$ we write $[\lambda]_{r \rightarrow s}$ for the bitstring $(\lambda_r, \lambda_{r+1}, \dots, \lambda_{s-1}, \lambda_s)$ —for $r = s$ it will be convenient to identify the bitstring (λ_r) with the individual bit λ_r . Finally, the concatenation of two tuples $\lambda = (\lambda_0, \dots, \lambda_m)$, $\mu = (\mu_0, \dots, \mu_n)$ will be denoted by $\lambda || \mu := (\lambda_0, \dots, \lambda_m, \mu_0, \dots, \mu_n)$.

Public key. The public key is the function $G : K^{37} \longrightarrow K^{26}$ defined by

$$G(X) = [t(\varphi^{-1}(F(\varphi(s(X)))))]_{0 \rightarrow 181}.$$

By construction $(Y_0, \dots, Y_{25}) = G(X_0, \dots, X_{36})$ can be expressed in the form

$$\begin{aligned} Y_0 &= P_0(X_0, \dots, X_{36}) \\ &\vdots \\ Y_{25} &= P_{25}(X_0, \dots, X_{36}) \end{aligned}$$

where each P_i is a polynomial of total degree ≤ 2 with coefficients in K' .

Note that the public key does not depend on the secret string Δ ; the latter is used in the actual signing procedure:

2.2 The Signing Algorithm

To compute the signature S of a bitstring M the following steps are performed (see [7] for a description of SHA-1):

1. Compute the 182-bit string

$$V := [\text{SHA-1}(M)]_{0 \rightarrow 159} \parallel [\text{SHA-1}(\text{SHA-1}(M))]_{0 \rightarrow 21}.$$

2. Compute the 77-bit string

$$W := [\text{SHA-1}(V \parallel \Delta)]_{0 \rightarrow 76}.$$

3. Compute

$$\begin{aligned} Y &:= (\pi([V]_{0 \rightarrow 6}), \pi([V]_{7 \rightarrow 13}), \dots, \pi([V]_{175 \rightarrow 181})) \in K^{26} \\ R &:= (\pi([W]_{0 \rightarrow 6}), \pi([W]_{7 \rightarrow 13}), \dots, \pi([W]_{70 \rightarrow 76})) \in K^{11} \end{aligned}$$

4. Compute

$$B := \varphi(t^{-1}(Y \parallel R)) \in L.$$

5. Compute

$$A := F^{-1}(B) \in L.$$

6. Compute

$$X = (X_0, \dots, X_{36}) := s^{-1}(\varphi^{-1}(A)) \in K^{37}.$$

7. Compute the signature S as

$$S := \pi^{-1}(X_0) \parallel \dots \parallel \pi^{-1}(X_{36}) \in \{0, 1\}^{259}.$$

We omit the description of the signature verification procedure, as our attack does not need it and reveals the complete secret key (Δ, s, t) ; an overview of the signing procedure is given in Figure [red square].

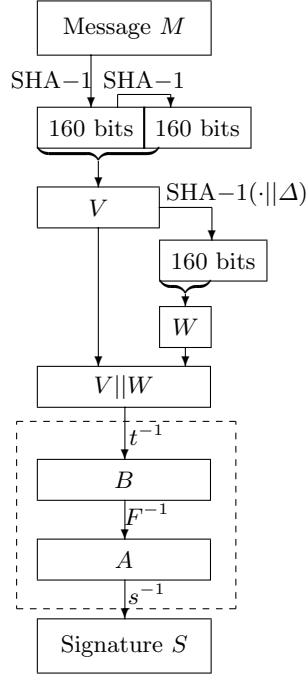


Fig. 1. Signature generation with FLASH and SFLASH.

3 Attacking SFLASH

As SFLASH has been proposed for the use on low-cost smartcards, we can assume that the attacker Eve has access to some smartcard implementing the above signature scheme and that Eve is also able to measure the power consumption of this smartcard during the execution of the signing algorithm. Following the established terminology of [2] we use the term *trace* to refer to a sequence of power consumption measurements taken during an execution of the signing algorithm.

Typically, the number of traces which have to be taken for performing a DPA is less than 1000, i. e., we have to measure the power consumption of several hundred executions of the signing algorithm. For sampling the power consumption at a rate of up to say 100 MHz no expensive specialized hardware equipment is needed (a typical clock rate of a smartcard is about 5 MHz). As we are not aware of any already available smartcard implementation of SFLASH (or FLASH), in the sequel we restrict to a theoretical description of our attack. Nevertheless we think the attack to be realistic and convincing, as its hardware-dependent aspects are not significantly different from already experimentally verified standard applications of DPA.

So in the sequel we assume that the attacker has sufficiently many traces of the smartcard's power consumption during the execution of the signing algorithm at her disposal; all other computations described in the sequel can be performed "off-line", i. e., the smartcard is not needed for them.

3.1 DPA Technique

For sake of completeness, in the sequel we give a short sketch of the main ideas involved when doing a DPA by means of a DPA selection function $D(M, b)$; a more detailed explanation can, e. g., be found in [2].

Given an input message M and some guess $b = (b_0, \dots, b_w)$ for a small part of the secret key, the DPA selection function $D(M, b)$ yields either zero or one. The idea is that for a correct guess b of the partial key, the value of $D(M, b)$ coincides with some bit processed in the execution of the signing algorithm with input M , and for an incorrect guess the value of $D(M, b)$ is correct with probability $\approx 1/2$.

Now, for $0 \leq \mu \leq m-1$ let $P_\mu = ((P_\mu)_0, \dots, (P_\mu)_{k-1})$ be a k -sample trace that describes the smartcard's power consumption during the execution of the signing algorithm with (known) input M_μ . Then for a fixed DPA selection function $D(M, b)$ we compute a k -sample *differential trace*

$$\delta = (\delta_0, \dots, \delta_{k-1});$$

namely, we determine the difference between the averages of the traces for which $D(M, b)$ evaluates to 1 and the average of the traces for which $D(M, b)$ is 0:

$$\delta_j := \frac{\sum_{\mu=0}^{m-1} D(M_\mu, b) \cdot (P_\mu)_j}{\sum_{\mu=0}^{m-1} D(M_\mu, b)} - \frac{\sum_{\mu=0}^{m-1} (1 - D(M_\mu, b)) \cdot (P_\mu)_j}{\sum_{\mu=0}^{m-1} (1 - D(M_\mu, b))} \quad (1)$$

(for $0 \leq j \leq k-1$). If our guess for the bits b is incorrect, then $D(M, b)$ is essentially uncorrelated to what was actually computed by the smartcard, and one can expect the values of the δ_j to be ≈ 0 for a sufficiently large number m of traces. So the plot of δ can be expected to be flat. On the other hand, if our guess for b is correct, then $D(M, b)$ is correlated to the corresponding intermediate result in the execution of the signing algorithm. And as the power consumption of the smartcard during the execution of the signing algorithm is correlated to the values of the bits involved in the computation, a significant peak in the plot of δ will reveal a correct guess of b .

Another (simpler) variant of DPA can be applied when all input values of some operation performed on the smartcard are known: say we know for some fixed $i \in \{0, \dots, 31\}$ the value of the bits m_i, b_i of the arguments $m = (m_0, \dots, m_{31}), b = (b_0, \dots, b_{31})$ of a 32-bit XOR operation where m_i is 0 for about half of the input messages M , and b_i is part of the secret key. Choosing a suitable threshold value $\sigma \in \mathbb{R}$ we can define a correlation function

¹ The precise value of k depends on the clock rate of the smartcard and the amount of time needed for signing, but one can expect $k < 2^{27}$.

$E(M, b_i)$ by setting $E(M, b_i) := 1$ if the expected power consumption of the XOR operation $m_i \oplus b_i$ for the input message M and the key bit value b_i is $> \sigma$, and $E(M, b_i) := 0$, otherwise. A suitable value of σ can, e. g., be found by means of a hardware simulation tool where we can determine the power consumption for all possible input values m_i, b_i . Next, we can define a differential trace as in (II) with $E(M, b_i)$ playing the role of $D(M, b)$. If our guess for b_i was correct, we can expect a significant positive peak in a plot of the corresponding differential trace. Such a peak reflects the correlation between the values of $E(M, b_i)$ and the actual power consumption of the smartcard during the execution of the signing algorithm.

On the other hand, if our guess for b_i was incorrect, we can expect a significant negative peak in a plot of the corresponding differential trace, because when we predict a high power consumption during the execution of the signing algorithm there typically is a low power consumption and vice versa. All the same, from the plot of the differential trace we can read off the correct value of the secret bit b_i .

The threshold-based DPA variant just described can also be applied if more bits b_i are to be guessed. In this case the negative peak usually becomes smaller or even vanishes. This case is described and used in Section 3.3.

3.2 Revealing Δ

For revealing Δ we first note that Step 1 of the signing algorithm can also be executed by Eve, as no secret information is needed here. So for any known message M , Eve can easily compute V , and up to the 80 bits of Δ she knows the complete input of the SHA-1 algorithm in Step 2. To see how to exploit this knowledge, we have to recall the first steps² of SHA-1 (cf. [7] for details):

1. First the padding scheme of SHA-1 yields the 512-bit string

$$A := V || \Delta || (1, \underbrace{0, \dots, 0}_{240 \text{ zeroes}}, 1, 0, 0, 0, 0, 0, 1, 1, 0).$$

2. Then A is split into sixteen 32-bit words W_0, \dots, W_{15} where W_0 is the left-most part of A . In particular W_0, \dots, W_4 depend only on V (and hence are known to Eve), W_5 contains 10 unknown bits (the left-most 10 bits of Δ), W_6 and W_7 are unknown, W_8 contains six unknown bits, and finally W_9, \dots, W_{15} are known.
3. Next, the following loop is executed (where S^1 stands for the circular left shift and \oplus for the XOR on 32-bit words):

$$\text{for } t \leftarrow 16 \text{ to } 79 \text{ do } W_t \leftarrow S^1(W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16})$$

² We remark that [7] does not prescribe all implementation details; for sake of concreteness we base our discussion on the computation method described in [7] Section 7]. Adapting the above approach to the procedure in [7] Section 8] is straightforward, as the latter uses XOR operations analogous to those exploited in our attack.

In particular for $t \in \{16, 20\}$ the right-hand side of the above assignment contains precisely one of the (partially) unknown words W_i :

$$W_{16} \leftarrow S^1(W_{13} \oplus \underline{W_8} \oplus W_2 \oplus W_0) \quad (2)$$

$$W_{20} \leftarrow S^1(W_{17} \oplus W_{12} \oplus \underline{W_6} \oplus W_4) \quad (3)$$

(where $W_{17} = S^1(W_{14} \oplus W_9 \oplus W_3 \oplus W_1)$ is known)

Similarly, for $t \in \{19, 23\}$ we obtain assignments depending only on known bits and W_{16} resp. W_{20} :

$$W_{19} \leftarrow S^1(W_{16} \oplus W_{11} \oplus \underline{W_5} \oplus W_3) \quad (\text{with } W_{16} \text{ as in (2)}) \quad (4)$$

$$W_{23} \leftarrow S^1(W_{20} \oplus W_{15} \oplus W_9 \oplus \underline{W_7}) \quad (\text{with } W_{20} \text{ as in (3)}) \quad (5)$$

We want to exploit these connections to reveal Δ “bit by bit” through a differential power analysis. For this we define for $0 \leq i \leq 79$ a function $e_i(M, b)$ which takes a bitstring M (the message to be signed) and a single bit b (bit no. i of Δ) as input. The output of $e_i(M, b)$ is the value of the corresponding bit in one of the XORs (2)–(5). More formally, we set

$$e_i(M, b) := \begin{cases} b \oplus [W_{16} \oplus W_{11} \oplus W_3]_{i+22 \rightarrow i+22}, & \text{for } 0 \leq i \leq 9 \\ b \oplus [W_{17} \oplus W_{12} \oplus W_4]_{i-10 \rightarrow i-10}, & \text{for } 10 \leq i \leq 41 \\ b \oplus [W_{20} \oplus W_{15} \oplus W_9]_{i-42 \rightarrow i-42}, & \text{for } 42 \leq i \leq 73 \\ b \oplus [W_{13} \oplus W_2 \oplus W_0]_{i-74 \rightarrow i-74}, & \text{for } 74 \leq i \leq 79 \end{cases}.$$

For a correct (incorrect) value of b the resulting value $e_i(M, b) \in \{0, 1\}$ is the (complement of the) value computed on input M during the execution of one of the assignments (2)–(5) in the signing algorithm. Moreover, as the words W_3 , W_4 , W_{20} , and W_0 are essentially determined by the SHA-1-hash value of the bitstring M , we can assume that for a randomly chosen input message M , the value “XORed with b ” in the definition of $e_i(M, b)$ equals 0 with probability $\approx 1/2$.

Finally, computing the value of $e_i(M, b)$ for arbitrary M and b provides no problem for $10 \leq i \leq 41$ and $74 \leq i \leq 79$. So for these values of i we have all the necessary ingredients for doing a “differential power analysis with known input values” as described in the second part of Section 3.1. Namely, we can base our threshold function $E_i(M, b)$ on the power consumption of the XOR operation that yields the value $e_i(M, b)$. The precise inputs of this XOR operation depend on the order in which the sum (2) resp. (3) is computed, but it can be assumed to be known to the attacker.

After having recovered the bits no. 10–41 and 74–79 of Δ successfully, we can proceed analogously with the $e_i(M, b)$ where $0 \leq i \leq 9$ resp. $42 \leq i \leq 73$.

3.3 Revealing the Affine Bijections s and t

In Step 4 of the signing algorithm the secret affine bijection t^{-1} is evaluated. Writing t in the form

$$\begin{aligned} t : K^{37} &\longrightarrow K^{37} \\ (b_0, \dots, b_{36}) &\longmapsto T_L \cdot (b_0, \dots, b_{36})^T + T_C, \end{aligned}$$

evaluating t^{-1} translates into evaluating

$$\begin{aligned} t^{-1} : K^{37} &\longrightarrow K^{37} \\ (b_0, \dots, b_{36}) &\longmapsto T_L^{-1} \cdot (b_0, \dots, b_{36})^T + T_L^{-1} \cdot T_C \end{aligned}$$

(recall that $\text{char}(K) = 2$).

In other words, for evaluating t^{-1} at $(b_0, \dots, b_{36}) \in K^{37}$ we have to perform one matrix-vector multiplication (with a matrix containing only $\{0, 1\}$ -entries) and one vector-addition. As the vector $T'_C := T_L^{-1} \cdot T_C$ also contains $\{0, 1\}$ -entries only, the latter can be implemented through 37 simultaneous XOR operations if the elements of K are represented in a polynomial basis. Having in mind smartcard applications, evaluating t^{-1} should not require large amounts of memory. Hence, a possible implementation for evaluating t^{-1} might look as follows:

Start by multiplying the first column of T_L^{-1} with b_0 and store this vector. As T_L^{-1} contains only $\{0, 1\}$ -entries, this multiplication can be realized (in parallel for the complete column) with AND gates. Next, multiply the second column of T_L^{-1} with b_1 and add the result to the stored vector. Using, e.g., a polynomial basis representation for K , this vector addition can be realized through $37 \cdot 7$ simultaneous XOR operations. We can proceed in the same manner with b_2, \dots, b_{36} and are finally left to add the constant vector T'_C which can be accomplished analogously.

By means of the traces of the smartcard's power consumption, it is possible to extract the secret parameter t from such an implementation.—For other “sensible” implementation approaches similar problems arise. Lacking an existing hardware implementation, for sake of concreteness here we stick to the above modeling.

To reveal t we proceed in two steps: first we recover the matrix T_L^{-1} and the vector T'_C up to a permutation of the rows. Then we make use of Step 5 of the signing algorithm for determining the correct order of the rows.

Determinining the rows. First we want to determine a set $Z_{0-12} \subseteq K'^{13}$ containing all those elements from K'^{13} which occur in the left-most 13 columns³ of T_L^{-1} . For this, assume that some guess $(b_0, \dots, b_{11}) \in K'^{12}$ has been fixed

³ One may also start with a different number of columns, say in the range 2–14. As long as the number of columns considered simultaneously is not too large, this has no significant influence on the required computational effort.

already, i. e., we think that there is some row index $0 \leq i \leq 36$ such that the equality

$$(b_0, \dots, b_{11}) = ((T_L^{-1})_{i,0}, \dots, (T_L^{-1})_{i,11})$$

holds. Then we can compute the sum $\sum_{j=0}^{11} b_j \cdot (Y||R)_j \in K$.—Note that the vector $(Y||R) = ((Y||R)_0, \dots, (Y||R)_{36}) \in K^{37}$ is known after having revealed Δ . In other words, if our guess (b_0, \dots, b_{11}) is correct, then we can compute one of the entries of a vector that is stored as intermediate result during the evaluation of t^{-1} . To this entry the product $(Y||R)_{12} \cdot (T_L^{-1})_{i,12}$ will be added during the execution of the signing algorithm, and we base our attack on the power consumption of this addition (seven simultaneous XOR operations): e. g., by means of a simulation we can determine for which pairs of elements from K^2 the XOR gates performing the latter addition have a high overall power consumption, and for which elements we can expect a low overall power consumption. Note that the implementation of the addition operation (seven simultaneous XORs) is identical for all row indices $0 \leq i \leq 36$, so we do not have to know the precise value of i .

Choosing a suitable threshold σ , we define the value of the correlation function $E_{0-12}(M, b)$ as 1 if for the plaintext M and

$$((T_L^{-1})_{i,0}, \dots, (T_L^{-1})_{i,12}) = \underbrace{(b_0, \dots, b_{12})}_{=b}$$

we have an overall power consumption $> \sigma$ for the corresponding XOR gates. For an overall power consumption $\leq \sigma$ we define $E_{0-12}(M, b)$ to be 0. As a typical value of σ we may think of the average value (taken over all possible inputs) of the overall power consumption of the XOR gates realizing the corresponding addition in K .

Now consider the differential trace $\delta_{0-12} := ((\delta_{i,0-12})_0, \dots, (\delta_{0-12})_{k-1})$ which is defined as follows:

$$(\delta_{0-12})_l := \frac{\sum_{\mu=0}^{m-1} E_{0-12}(M_\mu, b) \cdot (P_\mu)_l}{\sum_{\mu=0}^{m-1} E_{0-12}(M_\mu, b)} - \frac{\sum_{\mu=0}^{m-1} (1 - E_{0-12}(M_\mu, b)) \cdot (P_\mu)_l}{\sum_{\mu=0}^{m-1} (1 - E_{0-12}(M_\mu, b))}$$

(for $0 \leq l \leq k-1$). By construction, for a correct guess of b , i. e., for $b = ((T_L^{-1})_{i,0}, \dots, (T_L^{-1})_{i,12})$, we can expect significant peaks in a plot of δ_{0-12} .

As indicated above already, the differential trace δ_{0-12} cannot distinguish between the individual rows of T_L^{-1} , because the computations for the individual rows are identical (and performed in parallel). Consequently, we have significant peaks if our guess b coincides with $((T_L^{-1})_{i,0}, \dots, (T_L^{-1})_{i,12})$ for *any* $0 \leq i \leq 36$. On the other hand, for a choice of b that is different from the first 13 entries of *all* rows of T_L^{-1} we can expect a rather weak correlation between the values of $E_{0-12}(M, b)$ and the actual power consumption during the execution of the signing algorithm.—Of course, this reasoning implicitly assumes $b \neq (0, \dots, 0)$, as otherwise the value of $E_{0-12}(M, b)$ is independent of M , and δ_{0-12} is not well-defined.

In other words, for an incorrect guess $b \neq 0$ there are no significant peaks in a plot of δ_{0-12} , and by computing the plot of δ_{0-12} for all $b \in K'^{13} \setminus \{0\}$ we will get a set $Z_{0-12} \subseteq K'^{13}$ with ≤ 37 elements which contains all those elements $\neq 0$ that occur somewhere in the first 13 columns of T_L^{-1} . But we cannot expect to know the precise position or the number of occurrences of these words in the matrix T_L^{-1} . Finally, if Z_{0-12} is of cardinality < 37 we do not know whether for some row index the entries in the columns 0–12 all vanish, so in this case we add $0 \in K'^{13}$ to Z_{0-12} .

In the next step of the attack we want to modify Z_{0-12} in such a way that we obtain candidates for the first say 26 columns of T_L^{-1} . For this aim, we perform the following steps: for each $(b_0, \dots, b_{12}) \in Z_{0-12}$ we make guesses b_{13}, \dots, b_{24} for the matrix entries $(T_L^{-1})_{i,13}, \dots, (T_L^{-1})_{i,24}$ and compute the sum $\sum_{j=0}^{24} b_j \cdot (Y||R)_j \in K$.—Again, the precise value of i is not known to us, but it is sufficient for us to know that there is *some* row which starts with (b_0, \dots, b_{12}) .

This means that for a correct choice of (b_0, \dots, b_{24}) we compute the i^{th} entry of a vector that is stored as intermediate result during the evaluation of t^{-1} . To this entry the product $(Y||R)_{25} \cdot (T_L^{-1})_{i,25}$ will be added in the execution of the signing algorithm. Using guesses $b_{25} \in K'$ for $(T_L^{-1})_{i,25}$ and differential traces completely analogous to the above, in this way we obtain a set $Z_{0-25} \subseteq K'^{26}$ of cardinality ≤ 37 which contains all those elements from K'^{26} that occur somewhere in the first 26 columns of T_L^{-1} (possibly plus the zero vector $0 \in K'^{26}$). In particular, the number of differential traces to be computed for deriving Z_{0-25} from Z_{0-12} is $\leq 37 \cdot 2^{12+1} = 303, 104$.

Finally, in the same manner we go for the remaining 11 columns of T_L^{-1} and the constant vector T_C : for each $(b_0, \dots, b_{25}) \in Z_{0-25}$ we fix guesses $b_{26}, \dots, b_{36} \in K'$ for the entries in the last columns of T_L^{-1} and compute the sum $\sum_{j=0}^{36} b_j \cdot (Y||R)_j \in K$. Then using a guess $b_{37} \in K'$ for the corresponding entry of T'_C we look at differential traces to identify correct “completions” of the elements in Z_{0-25} . As the matrix T_L^{-1} is regular we do not have to take guesses with $(b_0, \dots, b_{36}) = (0, \dots, 0)$ into account here, and after computing $\leq 37 \cdot 2^{11+1}$ differential traces we will end up with a set $Z_{0-37} \subseteq K^{37} \times K$ of cardinality 37 containing the rows of T_L^{-1} plus the corresponding entry of T'_C . So we are left with the task of arranging these elements in the correct order, i. e., to find out which element in Z_{0-37} contains which “row of t^{-1} ”.

Sorting the rows. Knowing the affine bijection t^{-1} “up to permutation of the rows” we are able to determine for any given message M the output $B := \sum_{i=0}^{36} b_i X^i \in L$ of Step 4 of the signing algorithm up to permutation of the b_i .—By fixing some arbitrary order of the rows this permutation will be unknown but identical for all messages M . For deriving the correct permutation, we make use of Step 5 of the signing algorithm:

as explained in the specification [5] of SFLASH, the computation of $F^{-1}(B)$ can be realized by computing a suitable power of B with the square-and-multiply method. Further on, two alternate methods for computing $F^{-1}(B)$

are mentioned; the first one also involves the computation of a suitable power of B (namely B^{64}), and the second one is based on solving a system of linear equations over K . Here we focus on the first two suggestions for computing $F^{-1}(B)$ —for the third proposed method, the evaluation of the K -linear mapping $L \rightarrow L, \alpha \mapsto \alpha^{128^{11}}$ could serve as starting point for a DPA-based attack.

So assume that the smartcard has to compute a power B^h (with $h > 1$) of an element $B = \sum_{i=0}^{36} b_i X^i \in L$. Using square-and-multiply, for instance, we can expect that this computation will involve the computation of B^2 by means of some hardware multiplier. Having in mind the structure of a typical hardware multiplier (cf., e.g., [11]), for each $0 \leq i \leq 36$ such a computation will involve the 37 multiplications $b_i \cdot b_j$ ($0 \leq j \leq 36$) which are usually performed in parallel. In particular, for a fixed b_i the overall power consumption of the parallel multiplications $b_i \cdot b_j$ ($0 \leq j \leq 36$) is invariant under permutation of the “coefficients” b_j of B .

Knowing the coefficients of B up to permutation we can, e.g., by simulation, determine the overall power consumption of this computation for all 37 coefficients b_i ($0 \leq i \leq 36$). Hence, using some suitable threshold value τ we define a correlation function $E(M, j)$ as follows: if $B = \sum_{i=0}^{36} b_i X^i \in L$ is the result of Step 4 in the signing algorithm on input M , then for $0 \leq j \leq 36$ the value of $E(M, j)$ is defined to be 1 if the overall power consumption of the multiplications $b_j \cdot b_0, \dots, b_j \cdot b_{36}$ is $> \tau$. If the power consumption is $\leq \tau$, then we set $E(M, j) := 0$.

Next, we define a differential trace $\delta = (\delta_0, \dots, \delta_{k-1})$ by setting

$$\delta_l := \frac{\sum_{\mu=0}^{m-1} E(M_\mu, j) \cdot (P_\mu)_l}{\sum_{\mu=0}^{m-1} E(M_\mu, j)} - \frac{\sum_{\mu=0}^{m-1} (1 - E(M_\mu, j)) \cdot (P_\mu)_l}{\sum_{\mu=0}^{m-1} (1 - E(M_\mu, j))}$$

(for $0 \leq l \leq k-1$).

By construction we can expect a significant peak in a plot of δ for each choice $0 \leq j \leq 36$ of the second argument of $E(M, j)$. The position of such a peak indicates the time at which the parallel multiplications $b_j \cdot b_0, \dots, b_j \cdot b_{36}$ are executed. Of course, it can happen that not all of the coefficients of B are distinct, say the coefficient b_j occurs d times in B ; in this case the corresponding plot can be expected to have d significant peaks instead of only one.

All the same, the positions of the peaks in the plot of δ for the different choices of the second argument $0 \leq j \leq 36$ of $E(M, j)$ enable us to determine the order in which the coefficients of B are processed in the multiplication circuit. And of course, the order in which the coefficients of B are to be given as input to the hardware multiplier is known—typically one starts with b_{36} , and continues with b_{35}, \dots, b_0 . In summary, all we have to do for finding the correct order of the “rows of t^{-1} ” is to sort the 37 plots of δ (with the 37 possible choices $0 \leq j \leq 36$ for the second argument of $E(M, j)$) according to the position of the peaks.

Thereafter the affine bijection t^{-1} resp. t is revealed completely, and we are now able to mimick the signing algorithm up to and including Step 5. For determining the “rows of s^{-1} ” in Step 6 we can use precisely the same method

we used above to reveal t^{-1} . Determining the correct order of these rows is no problem, as the result of the evaluation of s^{-1} is the publicly known signature of the corresponding input message. So by looking at the signatures of some messages, we can simply read off the correct order of the rows.

In summary, for a smartcard implementation of SFLASH as discussed above, knowing sufficiently many traces of the smartcard's power consumption is enough to reveal the complete secret key. As it is unclear how a realistic DPA-resistant implementation of SFLASH should look like, we dissuade from using SFLASH on smartcards in the proposed form.

In the next section we shortly discuss the necessary adaptations of our attack when dealing with the signature scheme FLASH instead of SFLASH.

4 Adapting the Attack to FLASH

The structure of FLASH is almost identical to that of SFLASH; in particular the secret key consists again of a secret 80-bit string Δ and two affine bijections s and t . The only essential difference between FLASH and SFLASH is the size of the underlying parameters. E. g., instead of setting $K := \mathbb{F}_2[X]/(X^7 + X + 1)$, in FLASH the larger field $K := \mathbb{F}_2[X]/(X^8 + X^6 + X^5 + X + 1)$ (with a corresponding map π) is used. As our attack on SFLASH applies almost verbatimly to FLASH, in the sequel we restrict our attention to those aspects of the attack where some modifications of the techniques used for SFLASH are necessary.

4.1 Revealing Δ

In FLASH the secret 80 bit-string Δ also occurs as part of the argument of the SHA-1 algorithm. Namely, similarly as in SFLASH we have a statement of the form

$$W := [\text{SHA-1}(V \parallel \Delta)]_{0 \rightarrow 87}$$

where V is a known 208-bit string. So the padding scheme of SHA-1 yields the 512-bit string

$$A := V \parallel \Delta \parallel (\underbrace{1, 0, \dots, 0}_{214 \text{ zeroes}}, 1, 0, 0, 1, 0, 0, 0, 0, 0).$$

When splitting A into sixteen 32-bit words W_0, \dots, W_{15} (in the execution of SHA-1), the secret bits are distributed among the blocks W_6 , W_7 , and W_8 : while in W_6 only the “last” 16 bits are unknown, the blocks W_7 and W_8 are determined completely by the secret string Δ . All the same, for revealing Δ we can use precisely the same DPA-based attack as described for SFLASH in Section 3.2: first we use the assignments (2) and (3) to reveal W_8 and (the unknown 16 bits of) W_6 , and thereafter we reveal W_7 by means of the assignment (5).

4.2 Revealing the Affine Bijections s and t

While in SFLASH the entries of the matrices and vectors defining the affine bijections s and t are individual bits, in FLASH these matrices resp. vectors can contain arbitrary elements from K . So when implementing s^{-1} resp. t^{-1} it is not sufficient to use only AND operations for multiplying an element from K with a column of, say, the matrix T_L^{-1} . Instead a “full” multiplication in K has to be performed here—typically a linear feedback shift register (LFSR) is used for such a purpose. One might think of making use of this multiplication circuit in a DPA-based attack, but we can also proceed analogously as in SFLASH: because one matrix entry $(T_L^{-1})_{i,j}$ consists of 8 bits already, instead of going twice for 13 columns and once for 11+1 columns (as we did in SFLASH) this time it is sensible to go for each column individually.—Knowing the set of entries of the first $j - 1$ columns of T_L^{-1} already, we look at the power consumption of the addition of $(Y||R)_{j-1} \cdot (T_L^{-1})_{i,j-1}$ to the intermediate result.

With a polynomial basis representation, in FLASH such an addition in K corresponds to 8 simultaneous XOR operations. It is worth remarking here, that the first column usually does not have to be treated separately, although there is no intermediate result present when this column is processed: for storing as intermediate result the product of $(Y||R)_0$ with the first column, typically an addition of that product to the zero vector will be implemented, i. e., we have $37 \cdot 8$ simultaneous XOR operations again.

To find the correct order of the “rows of t^{-1} ” we can use the same technique as for SFLASH. Also determining s^{-1} does not require any new ideas—first we go for the set of rows as described above for t^{-1} , and then we determine the correct order by simply looking at some signatures.

In summary, a smartcard implementation of FLASH suffers from essentially the same problems as a smartcard implementation of SFLASH. In fact, owing to the K -multiplications (with LFSRs) in the evaluation of t^{-1} resp. s^{-1} , FLASH seems to be even more susceptible to DPA-based attacks than SFLASH. Consequently, as we are not aware of any suggestions for realistic DPA-resistant smartcard implementations of FLASH, we also dissuade from using the NESSIE candidate FLASH in the proposed form.

5 Conclusion

We have demonstrated that the design of the NESSIE candidates FLASH and SFLASH does not provide sufficient resistance against DPA-based attacks. In the contrary, under certain assumptions about the underlying implementation, this kind of attack can be used to reveal the complete secret key.

As it is unclear how a DPA-resistant implementation of these signature schemes should look like, in the present form these NESSIE candidates cannot be considered as sufficiently secure for cryptographic smartcard applications.

Acknowledgements. We would like to thank one of the anonymous reviewers for her/his helpful comments.

References

1. T. BETH AND D. GOLLMANN, *Algorithm Engineering for Public Key Algorithms*, IEEE Journal on selected areas in communications, 7 (1989), pp. 458–466.
2. P. KOCHER, J. JAFFE, AND B. JUN, *Differential Power Analysis*, in *Advances in Cryptology—CRYPTO '99*, M. Wiener, ed., vol. 1666 of *Lecture Notes in Computer Science*, Springer, 1999, pp. 388–397.
3. J. PATARIN, N. COURTOIS, AND L. GOUBIN, C^*_{-+} and HM : *Variations around two schemes of T. Matsumoto and H. Imai*, in *Advances in Cryptology—ASIACRYPT '98*, K. Ohta, ed., vol. 1514 of *Lecture Notes in Computer Science*, Berlin, 1998, Springer.
4. ———, *FLASH, a fast asymmetric signature scheme for low-cost smartcards. Primitive specification and supporting documentation*. Presented at First Open NESSIE Workshop, November 2000. At the time of writing available electronically at the URL <https://www.cosic.esat.kuleuven.ac.be/nessie/workshop/submissions/flash.zip>.
5. ———, *SFLASH, a fast asymmetric signature scheme for low-cost smartcards. Primitive specification and supporting documentation*. Presented at First Open NESSIE Workshop, November 2000. At the time of writing available electronically at the URL <https://www.cosic.esat.kuleuven.ac.be/nessie/workshop/submissions/sflash.zip>.
6. ———, *FLASH, a Fast Multivariate Signature Algorithm*, in *Progress in Cryptology — CT-RSA 2001*, D. Naccache, ed., vol. 2020 of *Lecture Notes in Computer Science*, Berlin; Heidelberg, 2001, Springer, pp. 298–307.
7. U.S. DEPARTMENT OF COMMERCE, NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY, *FIPS PUB 180-1 SECURE HASH STANDARD*, April 1995. At the time of writing available electronically at the URL <http://csrc.nist.gov/publications/fips/fips180-1/fips180-1.pdf>.

Quadratic Relations for S-Boxes: Their Minimum Representations and Bounds

Routo Terada and Paulo G. Pinheiro

University of S. Paulo, Brazil
`{rt, gerald}@ime.usp.br`

Abstract. We introduce polynomial approximations and consider the particular case of quadratic approximations. We establish an isomorphism between the set of quadratic Boolean functions and graphs. As its consequence, we can reduce problems involving quadratic Boolean functions into problems with graphs and vice-versa. We present the problem of finding a minimum representation of quadratic functions, and prove bounds on the number of terms and variables. With these bounds, we were able to find quadratic relations with the highest probabilities for SERPENT and CRYPTON, former AES candidates.

1 Introduction

Biham and Shamir introduced Differential Cryptanalysis [1][2][4], one of the most effective methods of measuring the strength of a block ciphering function. More recently another method was introduced by Matsui: Linear Cryptanalysis – LC for short – [3].

We define polynomial approximations for block ciphering functions, and their probabilities to occur. It is known LC gives only one bit of information with respect to the subkey bits occurring in the linear approximation. Motivated by LC, we consider polynomial approximations of order k in n variables, $k \leq n$. In the particular case of quadratic approximation, (i.e., $k = 2$), we show there is a useful isomorphism between the problem of determining μ and the problem of covering a graph of order k with a minimum number of complete bipartite subgraphs (which is well known to be an NP-complete problem). Using this isomorphism, we show the minimum number μ of order k terms, needed to represent these approximations, is at most $\lfloor \frac{n}{k} \rfloor$, and that the number of bits of information given by the approximation is at least $2^{k\mu}$. As a consequence of these results, we were able to find quadratic approximations for the former AES candidates SERPENT [5] and CRYPTON [6][7]. As stated in [9] (p. 3), “If we have a probabilistic relation between plaintext and ciphertext expressed as a polynomial, then we already have an information leakage and the cipher may be considered broken”.

Among previous related works, we should mention Moriai, Shimoyama and Kaneko [10], and Jakobsen [9]; they presented distinct approaches to nonlinear approximations for S-boxes, but either (1) the computational complexity of their

searching algorithms are much higher than ours, or (2) the relations they found do not have the highest probabilistic deviation (i.e., $1/2$), as ours do.

In Section (2) we formalize the concept of polynomial approximations, their probabilities to occur, and some lemmas. In Sections (3) and (4) we consider the particular case of quadratic approximations, and whether it's feasible to compute them or not; we illustrate the quadratic case with SERPENT. In Section (5), we establish an isomorphism between the set of quadratic Boolean functions and graphs. In Sections (6), (7) and (8), we present the problem of finding a minimum representation of quadratic functions, prove some properties and bounds on the number of terms and variables. In Section (9), we have additional theorems on graph coverings. Finally, in Section (10) we present quadratic relations for SERPENT and CRYPTON.

2 Basic Definitions and Results for Any Order

Let $\#S$ denote the cardinality of a set S . Let $GF(2)^n$ be the polynomials of order (or degree) n over $GF(2)$. Let $Q : GF(2)^n \rightarrow GF(2)^m$ be a function, and denote $y = Q(x)$. Let $\Psi_{in} : GF(2)^n \rightarrow GF(2)$, $\Psi_{out} : GF(2)^m \rightarrow GF(2)$, $x = (x_{n-1}, x_{n-2}, \dots, x_1 x_0)$, $y = (y_{m-1}, y_{m-2}, \dots, y_1 y_0)$.

Function Q can be approximated by a relation $\Psi(x, y) = (\Psi_{in}(x), \Psi_{out}(y))$ with probability $P_\Psi = \frac{\#\{x \in GF(2)^n \mid \Psi_{in}(x) = \Psi_{out}(Q(x))\}}{2^n}$.

Given a relation $\Psi = (\Psi_{in}, \Psi_{out})$, define *order* k of Ψ associated to the function Q as $k = \max\{\text{order}(\Psi_{in}), \text{order}(\Psi_{out})\}$.

The set of all relations of order k associated to a function Q in $GF(2)^n$ is denoted $\{\Psi\}_Q^k$.

Define $\delta = |P_\Psi - \frac{1}{2}|$, the *deviation* of P_Ψ from $\frac{1}{2}$. The set of all relations of order k with distance δ associated to Q is denoted $\{\Psi; \delta\}_Q^k$ and can write $\{\Psi\}_Q^k = \bigcup_{0 \leq \delta \leq \frac{1}{2}} \{\Psi; \delta\}_Q^k$.

From the definition of δ we can conclude not all values of δ are possible; by the following lemma, only a finite number of δ values may occur for a given Q .

Lemma 1. *Given $Q : GF(2)^n \rightarrow GF(2)^m$ and $\Lambda \in \{\Psi\}_Q^k$, the number $2^n \delta_\Lambda$ is a nonnegative integer in $[0, 2^{n-1}]$. In particular if $n > 2$ and Q is a permutation, then $2^n \delta_\Lambda$ is an even positive integer.*

We have proven the following lemmas.

Lemma 2. *Let $Q : GF(2)^n \rightarrow GF(2)^m$, $n > 2$. Then Q is a permutation iff for all relations $\Lambda \in \{\Psi\}_Q^k$, the number $2^n \delta_\Lambda$ is an even positive integer.*

Lemma 3. *Let $Q : GF(2)^n \rightarrow GF(2)^m$ be a permutation and $\text{order}_{\max}(Q) = k$. Then $\{\Psi; \frac{1}{2}\}_Q^k$ is a vector space over $GF(2)$ with the usual operations of (1) addition of relations and (2) multiplication of a relation by a scalar, and*

$$\#\{\Psi; \frac{1}{2}\}_Q^k = 2^{\binom{n}{k}}.$$

Lemma 4. *Let Q and R be permutations over $GF(2)^n$. Then $\#\{\Psi; \delta\}_Q^k = \#\{\Psi; \delta\}_R^k$ for all $\delta \in [0, \frac{1}{2}]$.*

Let $F[n; k]$ denote the set of all Boolean functions of order k over $GF(2)^n$. Let $F[n; \leq k]$ denote the subset of $F[n; k]$ with order $\leq k$, and $F[n; \geq k]$, with order $\geq k$.

For example, $F[2; 1] = \{x_0, x_1, x_0 + 1, x_1 + 1, x_0 + x_1, x_0 + x_1 + 1\}$.

Let $F^c[n; k]$ denote the set of all Boolean functions of order k over $GF(2)^n$ without the constant term. Let $F^c[n; \leq k]$ denote the subset of $F^c[n; k]$ with order $\leq k$, and $F^c[n; \geq k]$, with order $\geq k$.

Let $+$ and $*$ denote the usual addition and multiplication, respectively, of Boolean functions over $GF(2)^n$. Then we have the following:

Lemma 5. *For $n > 0, k \geq 0$, $(F[n; \leq k], +)$ is closed under $+$. Likewise, $(F[n; \geq k], *)$ is closed under $*$.*

Lemma 6. *$(F[n; \leq k], +)$ is a vector space over $GF(2)$, for $n > 0, k \geq 0$, with $\dim(F[n; \leq k]) = \sum_{i=0}^k \binom{n}{i}$.*

Lemma 7. *$(F[GF(2)^n; \leq n], +, *)$ is a commutative linear algebra with unity element over $GF(2)$.*

3 Quadratic Relations and Approximations

We consider the particular case of polynomial approximations of degree 2, i.e., quadratic approximations. This case is very interesting since, as we'll see later, we prove there is a one-to-one mapping between the set of quadratic Boolean functions in $GF(2)^n$ and the graphs of order n . As a consequence, we can reduce problems involving quadratic Boolean functions into problems with graphs and vice-versa. This way, we are able to obtain some bounds for the minimum representation of a quadratic Boolean function. Furthermore, we obtain a bound for the minimum number of bipartite complete graphs covering a graph of order n (the problem of determining this number is NP-complete) as a function of the minimum number of quadratic terms needed to represent a Boolean function associated to the graph.

4 Quadratic Relations

Let Q be a function from $GF(2)^n$ into $GF(2)^m$. The computation of all quadratic relations associated to Q is, in general, unfeasible even for relatively small values of m and n . The total number of quadratic relations associated to Q , i.e., $\#\{\Psi\}_Q^2$,

is $2^{\binom{n}{2} + \binom{m}{2} + n + m}$. If we consider S-boxes of block ciphers (which are our

main interest) the values of m and n are usually either the same or very close. So, if we consider $m = n$ the total number of quadratic relations associated to Q is 2^{n^2+n} . For $n \geq 6$ the computation of the set $\{\Psi\}_Q^2$ is still unfeasible. For $n = 6$, the required time, $O(2^{42})$, is feasible, but the storing of 2^{42} relations is not. An alternative strategy, to make the computation feasible, is to fix an S-box input relation (resp., output relation) and vary only the output (resp., input). In this case, the computational effort is reduced to $2^{\frac{n^2+n}{2}}$. This way, it's feasible to compute the quadratic relations for up to $n = 8$ – the required time is $O(2^{36})$. Since we are interested in relations with the highest probabilities, the space required to store these relations is negligible.

For SERPENT, this strategy can be applied since $n = m = 4$ for its eight S-boxes. But for CRYPTON, there are two 8×8 S-boxes and the computation of all the quadratic relations for each S-box takes time $O(2^{72})$. On the other hand, CRYPTON S-boxes have a Feistel structure with three iterations which use three 4×4 S-boxes, for which it's feasible to compute all the quadratic relations.

Definition 1. *One term of degree g of a Boolean relation is simple if it is a product of g variables.*

We'll now see how to represent a quadratic Boolean function as an integer vector.

- Let $Q : GF(2)^n \longrightarrow GF(2)^m$. If $\Lambda \in \{\Psi\}_Q^2$ then Λ has up to $\frac{n^2+m^2-(n+m)}{2}$ quadratic terms and $n + m$ linear terms. As we're interested in deviation of relations, we do not need to consider the independent term.

- Let

- B_{in}^2 be a permutation of all the simple quadratic terms with n variables;
- B_{out}^2 be a permutation of all the simple quadratic terms with m variables;
- B_{in}^1 be a permutation of all the simple linear terms with n variables;
- B_{out}^1 be a permutation of all the simple linear terms with n variables.

- This way, $B_{in}^2 B_{in}^1$ is a base for $F^c[n; \leq 2]$ and $B_{out}^2 B_{out}^1$ is a base for $F^c[m; \leq 2]$, where the concatenation means union of the two sets preserving the order of the elements.

- As $\Lambda_{in} \in F^c[n; \leq 2]$, we can write it as a vector (of 0 s and 1s) with respect to the base $B_{in}^2 B_{in}^1$. Let $\Lambda_{in} |_{B_{in}^2}$ be the integer corresponding to the binary number representation of the permutation B_{in}^2 , and let $\Lambda_{in} |_{B_{in}^1}$ be the integer corresponding to the binary number representation of the permutation B_{in}^1 .

- Similarly, let $\Lambda_{out} |_{B_{out}^2}$ be the integer corresponding to the binary number representation of the permutation B_{out}^2 , and let $\Lambda_{out} |_{B_{out}^1}$ be the integer corresponding to the binary number representation of the permutation B_{out}^1 .

- Then we can represent Λ uniquely by the 4-tuple

$$(\Lambda_{in} |_{B_{in}^2}, \Lambda_{in} |_{B_{in}^1}, \Lambda_{out} |_{B_{out}^2}, \Lambda_{out} |_{B_{out}^1})$$

with respect to the bases $B_{in}^2 B_{in}^1$ and $B_{out}^2 B_{out}^1$. The uniqueness is a consequence of the fact that $B_{in}^2 B_{in}^1$ and $B_{out}^2 B_{out}^1$ are bases of $F^c[n; \leq 2]$ and

$F^c[m; \leq 2]$, respectively. Observe that $0 \leq \Lambda_{in} |_{B_{in}^2} < 2^{\frac{n^2-n}{2}}$, $0 \leq \Lambda_{in} |_{B_{in}^1} < 2^n$, $0 \leq \Lambda_{out} |_{B_{out}^2} < 2^{\frac{m^2-m}{2}}$ and $0 \leq \Lambda_{out} |_{B_{out}^1} < 2^m$.

For example, consider the relation Λ :

$$x_3x_1 + x_3x_0 + x_1x_0 + x_1 + x_0 = y_3 + y_1 + y_0 \quad (1)$$

of the SERPENT S-box $S_0 : GF(2)^4 \rightarrow GF(2)^4$ which is valid with deviation $\delta_\Lambda = \frac{1}{2}$.

Let:

$$\begin{aligned} - B_{in}^2 &= \{x_3x_2, x_3x_1, x_3x_0, x_2x_1, x_2x_0, x_1x_0\}, \\ - B_{in}^1 &= \{x_3, x_2, x_1, x_0\}, \\ - B_{out}^2 &= \{y_3y_2, y_3y_1, y_3y_0, y_2y_1, y_2y_0, y_1y_0\}, \\ - B_{out}^1 &= \{y_3, y_2, y_1, y_0\}. \end{aligned}$$

Then:

$$\Lambda_{in} = (0, 1, 1, 0, 0, 1, 0, 0, 1, 1)_{B_{in}^2 B_{in}^1}$$

and

$$\Lambda_{out} = (0, 0, 0, 0, 0, 0, 1, 0, 1, 1)_{B_{out}^2 B_{out}^1}.$$

Thus, Λ can be numerically represented with respect to the bases $B_{in}^2 B_{in}^1$ and $B_{out}^2 B_{out}^1$ by the 4-tuple:

$$(25, 3, 0, 11).$$

Using this integer vector representation of quadratic Boolean relations, it is more feasible to search for the quadratic relations of a given function $Q : GF(2)^n \rightarrow GF(2)^m$. Such representation can be generalized for $\Lambda \in \{\Psi\}_Q^g$.

5 Quadratic Boolean Functions and Graphs

In this Section we show a fundamental relation between graphs and quadratic Boolean approximations.

We consider only the functions in $F^c[n; \leq 2]$, i.e., we're not interested in independent terms.

Theorem 1. *Let G^n be the set of all the graphs with order n . Then, there is a bijection between G^n and $F^c[n; \leq 2]$.*

Proof: Let B be a canonical base for $F^c[n; \leq 2]$. Let $\Phi : F^c[n; \leq 2] \rightarrow G^n$ which associates each element f in $F^c[n; \leq 2]$ to a graph G_f in G^n as follows:

$$\begin{aligned} - \Phi : F^c[n; \leq 2] &\rightarrow G^n \\ - \Phi(f) &= G_f \end{aligned}$$

- $\Phi = (\varphi_1, \varphi_2)$
- $\varphi_1 : \{0, 1, \dots, n-1\} \rightarrow V(G_f)$ (set of vertices)
- $\varphi_2 : f_B \rightarrow E(G_f)$ (set of edges)
- φ_1 associates each variable $x_i, i \in \{0, 1, \dots, n-1\}$, to the vertex $v_i \in V(G_f)$.
- Let f_B be the representation of f with respect to the canonical base B , and let $f_B(ij)$ be the coordinate of f for the term $x_i x_j$ and let $f_B(i)$ be the coordinate of f for the term $x_i, 0 \leq i < n$ and $0 \leq j < i$.
- φ_2 associates each $f_B(ij)$ to the edge $v_i v_j \in E(G_f)$ if $f_B(ij) = 1$, and each $f_B(i)$ to the loop $v_i v_i \in E(G_f)$ if $f_B(i) = 1$.

Clearly, φ_1 and φ_2 are invertible.

Hence, Φ is a bijection between $F^c[n; \leq 2]$ and G^n . \diamond

For example, consider the function $f \in F^c[4; \leq 2]$ defined by:

$$f(x) = x_3 x_1 + x_3 x_0 + x_1 x_0 + x_1 + x_0. \quad (2)$$

Then, the graph $G_f = \Phi(f) = (\varphi_1(\{0, 1, 2, 3\}), \varphi_2(f_B)) = (V(G_f), E(G_f))$ is defined as

$$\begin{aligned} V(G_f) &= \{v_0, v_1, v_2, v_3\} \\ E(G_f) &= \{\{v_0, v_1\}, \{v_0, v_3\}, \{v_1, v_3\}, \{v_0, v_0\}, \{v_1, v_1\}\}. \end{aligned}$$

Since an approximation Λ with degree 2 for a function Q is a parity test between two quadratic Boolean functions, we can represent it by a pair of graphs $G_{\Lambda_{in}}$ and $G_{\Lambda_{out}}$. Thus, whenever convenient, we can alternately use either f or G_f , in the statements to follow.

6 Minimum Forms of Quadratic Functions

By Theorem (II) we'll have a useful interpretation for the Minimum Form (or Representation) of Quadratic Boolean Functions.

Problem of Minimum Form of Quadratic Boolean Functions: Let $f \in F^c[n; \leq 2]$. Find a base B for $F^c[n; \leq 2]$ such that the number of quadratic terms in f_B is minimum.

Denote such f_B as $\mu(f)$ and denote the number of quadratic terms in f_B as $\#\mu(f)$. By Theorem (II), we may refer to them as $\mu(G_f)$ and $\#\mu(G_f)$, respectively.

For example, consider $f \in F^c[4; \leq 2]$ defined as:

$$f(x) = x_3 x_2 + x_3 x_1 + x_3 x_0 + x_2 x_0 + x_1 x_0 + x_3 + x_2. \quad (3)$$

The graph associated to f , G_f , is:

$$\begin{aligned} V(G_f) &= V(K_4) \\ E(G_f) &= E(K_4) \setminus \{\{v_1, v_2\}\} \cup \{\{v_2, v_2\}, \{v_3, v_3\}\}. \end{aligned}$$

The function f can be written as:

$$(x_3 + x_0)(x_3 + x_2 + x_1) + x_2. \quad (4)$$

Since there is at least one quadratic term, it follows that the form (4) is a minimum form for f . Thus,

$$\mu(f) = (x_3 + x_0)(x_3 + x_2 + x_1) + x_2$$

and $\# \mu(f) = 1$.

A base B such that $\# f_B = 1$ is:

$$B = \{(x_3 + x_0)(x_3 + x_2 + x_1), x_0x_1, x_1x_2, x_1x_3, x_2x_0, x_3x_0, x_3, x_2, x_1, x_0\}.$$

and

$$f_B = (1, 0, 0, 0, 0, 0, 0, 1, 0, 0)_B,$$

whose quadratic part is $(1, 0, 0, 0, 0, 0)$.

Definition 2. Let g and h be linear Boolean functions. Then, $\text{Var}(g)$ is the set of variables for g , $\|g\| = \#\text{Var}(g)$ and $g \cap h = \{a : a \in \text{Var}(g) \wedge a \in \text{Var}(h)\}$.

Definition 3. Consider the quadratic term $g(x)h(x)$, such that $g, h \in F^c[n; 1]$. We say gh is pure if $g \cap h = \emptyset$. Otherwise, we say gh is not pure.

Lemma 8. Let $f \in F^c[n; \leq 2]$. f is a pure quadratic term iff G_f is a complete bipartite graph.

Proof: Let $g, h \in F^c[n; 1]$.

(\implies) If $f = gh$ is a pure quadratic term, by Theorem (II) G_f is bipartite, since the subgraphs G_g and G_h are independent sets in $V(G_f)$ and $V(G_f) = V(G_g) \cup V(G_h)$. Since in the product of g and h all possible combinations of variables in g and h are generated (since f is pure), it follows that $E(G_f)$ is formed by all the possible edges from $V(G_g)$ to $V(G_h)$. Hence, G_f is a complete bipartite graph.

(\impliedby) If G_f is a complete bipartite graph with independent sets G_g and G_h , it follows that $f = gh$ is a pure quadratic term. Suppose the opposite is true, i.e., f is not pure. Then there is at least one variable a which belongs to $g \cap h$. Let $b \in \text{Var}(g) \setminus g \cap h$ and $c \in \text{Var}(h) \setminus g \cap h$. Under these conditions, such variables must exist since, otherwise, either $\text{Var}(g) \subseteq \text{Var}(h)$ or $\text{Var}(h) \subseteq \text{Var}(g)$. As $GF(2)$ is idempotent, we'd have $f = g$ or $f = h$, i.e., $\text{Ord}(f) = 1$, which contradicts the hypothesis. Therefore, the variables b and c must exist. Then, by Theorem (II), $\{a, b\} \in E(G_f)$ and $\{a, c\} \in E(G_f)$, but a and c are in $V(G_g)$, which contradicts the hypothesis that G_g is an independent set (absurd!). Thus, f is a pure quadratic term. Therefore, f is a pure quadratic term iff G_f is a complete bipartite graph. \diamond

Lemma 9. *Let $f \in F^c[n; \leq 2]$. f is not a pure quadratic term iff G_f is a complete tripartite graph such that all the vertices in one of the partitions have loops.*

Proof : Let $g, h \in F^c[n; 1]$.

(\Rightarrow) If $f = gh$ is not a quadratic term, then by Theorem (1), G_f is tripartite, since the subgraphs $G_{g \setminus g \cap h}$, $G_{h \setminus g \cap h}$ and $G_{g \cap h}$ are independent sets in $V(G_f)$ and $V(G_f) = V(G_{g \setminus g \cap h}) \cup V(G_{h \setminus g \cap h}) \cup V(G_{g \cap h})$. Then, as f is not pure, all the possible combinations in the product of g and h are generated, except the combinations of variables distinct from $g \cap h$ (which cancel each other due to the cancellation property of the addition in $GF(2)$) and all the vertices in $G_{g \cap h}$ have loops. Thus, G_f is a complete tripartite graph.

(\Leftarrow) If G_f is a complete tripartite graph with independent sets G_g , G_h and G_i such that all the vertices in G_i have loops, then $f = (g + i)(h + i)$ is not a pure quadratic term, as a direct consequence of Theorem (1) and Definition (3). Therefore, this lemma follows. \diamond

In the following sections, we show some upper- and lower-bounds for the number of quadratic terms and for the number of variables of the Minimum Form of Quadratic Boolean Functions.

7 Bounds for the Quadratic Terms (Edges)

Theorem 2. *If G is a graph of order n and $\#\mu(G) = 1$ then:*

$$\#E(G) \leq \lfloor \frac{n^2 + n + 1}{3} \rfloor.$$

Theorem 3. *If G is a graph of order n , $\#\mu(G) = 1$ and $\#E(G) = \lfloor \frac{n^2 + n + 1}{3} \rfloor$ then the number of loops in G is at least $\lfloor \frac{n+1}{3} \rfloor$ and at most $\lceil \frac{n+2}{3} \rceil$. In addition, there is always G that satisfies these hypotheses with $\lceil \frac{n+1}{3} \rceil$ loops.*

As consequences of Theorems (2) and (3) and their proofs, we'll state some corollaries which will be used in other proofs.

Corollary 1. *If G is a graph of order n , $(n + 2) \bmod 3 = 0$, $\#\mu(G) = 1$ and $\#E(G) = \lfloor \frac{n^2 + n + 1}{3} \rfloor$ then the number of loops in G is $\lceil \frac{n+1}{3} \rceil$.*

Corollary 2. *If G is a graph of order n , $(n + 1) \bmod 3 = 0$, $\#\mu(G) = 1$ and $\#E(G) = \lfloor \frac{n^2 + n + 1}{3} \rfloor$ then the number of loops in G is $\lceil \frac{n+1}{3} \rceil$ or $\lceil \frac{n+2}{3} \rceil$.*

Corollary 3. *If G is a graph of order n , $n \bmod 3 = 0$, $\#\mu(G) = 1$ and $\#E(G) = \lfloor \frac{n^2 + n + 1}{3} \rfloor$ then the number of loops in G is $\lfloor \frac{n+1}{3} \rfloor$ or $\lceil \frac{n+1}{3} \rceil$.*

8 Bounds for the Number of Variables (Vertices)

Theorem 4. $\#\mu(K_{m,n}) = 1$.

Proof : Since $K_{m,n}$ is a complete bipartite graph, such that one partition has m vertices and the other one has n vertices, it follows from Lemma (8) that $K_{m,n}$ can be written as a unique quadratic term. \diamond

Next, we'll prove some lemmas and theorems to be used in the proof of an exact value for $\#\mu(K_n)$.

Lemma 10. *If G is a complete graph of order n , $n \bmod 2 = 1$, then G can be represented by:*

$$K_n = \sum_{j=1}^{\frac{n-1}{2}} ((v_{2j} + v_{2j-1}) (\sum_{i=0}^{2j-1} v_i) + v_{2j-1}).$$

Lemma 11. *If G is a complete graph of order n , $n \bmod 2 = 0$, then G can be represented by:*

$$K_n = v_{n-1} (\sum_{i=0}^{n-2} v_i) + \sum_{j=1}^{\frac{n-2}{2}} ((v_{2j} + v_{2j-1}) (\sum_{i=0}^{2j-1} v_i) + v_{2j-1}).$$

Lemma 12. *If G is a complete graph of order n , $n \bmod 2 = 1$, then G can be represented by:*

$$K_n = \sum_{j=1}^{\frac{n-1}{2}} ((v_{2j} + v_{2j-1}) (\sum_{i=0}^{2j-1} v_i) + v_{2j-1}) \quad (5)$$

and this representation using $\frac{n-1}{2}$ quadratic terms is unique up to an isomorphism.

Lemma 13. *If G is a complete graph of order n , $n \bmod 2 = 0$, then G can be represented by:*

$$K_n = v_{n-1} (\sum_{i=0}^{n-2} v_i) + \sum_{j=1}^{\frac{n-2}{2}} ((v_{2j} + v_{2j-1}) (\sum_{i=0}^{2j-1} v_i) + v_{2j-1}) \quad (6)$$

and this representation using $\frac{n}{2}$ quadratic terms is unique up to an isomorphism.

Theorem 5. *If G is a complete graph of order n , then G can be represented by:*

- a) $K_n = \sum_{j=1}^{\frac{n-1}{2}} ((v_{2j} + v_{2j-1}) (\sum_{i=0}^{2j-1} v_i) + v_{2j-1})$, if n is odd,
- b) $K_n = v_{n-1} (\sum_{i=0}^{n-2} v_i) + \sum_{j=1}^{\frac{n-2}{2}} ((v_{2j} + v_{2j-1}) (\sum_{i=0}^{2j-1} v_i) + v_{2j-1})$, if n is even.

Such representation is unique, up to an isomorphism, with this number of quadratic terms.

Proof: It follows from Lemmas (I0) (I2), and (I3), that G can be represented by this form. \diamond

Theorem 6. $\# \mu(K_n) = \lfloor \frac{n}{2} \rfloor$.

Proof: (by induction on n) Let v_0, v_1, \dots, v_{n-1} be the vertices of K_n .

Base ($n = 1$): Clearly, this theorem is valid, since $\# \mu(K_1) = \lfloor \frac{1}{2} \rfloor = 0$ and K_1 has no edges.

Base ($n = 2$): Clearly, this theorem is valid, since $\# \mu(K_2) = \lfloor \frac{2}{2} \rfloor = 1$ and we need one quadratic term to represent $\{v_0, v_1\}$, namely, $\mu(K_2) = v_0 v_1$.

Base ($n = 3$): This theorem is valid since $E(K_3) = \{\{v_0, v_1\}, \{v_0, v_2\}, \{v_1, v_2\}\}$ which can be written as:

$$K_3 = (v_2 + v_1)(v_0 + v_1) = v_1 + v_0 v_2 + v_0 v_1 + v_1 v_2.$$

Thus, $\# \mu(K_3) = \lfloor \frac{3}{2} \rfloor = 1$.

Base ($n = 4$): This theorem is valid since

$$E(K_4) = \{\{v_0, v_1\}, \{v_0, v_2\}, \{v_0, v_3\}, \{v_1, v_2\}, \{v_1, v_3\}, \{v_2, v_3\}\}$$

which can be written as:

$$K_4 = (v_2 + v_1)(v_0 + v_1) + v_3(v_0 + v_1 + v_2) \quad (7)$$

$$K_4 = v_2 + v_0 v_2 + v_0 v_1 + v_0 v_3 + v_1 v_2 + v_1 v_3 + v_2 v_3.$$

Suppose $\# \mu(K_4) = 1$. It follows from Theorem (2), if $\# \mu(K_4) = 1$, then $\# E(K_4) \leq 7$, but, by Corollary (I), the number of loops of K_4 if $\# \mu(K_4) = 1$ is 2. Thus, the number of edges of K_4 , which are not loops is 5, and can be represented by a unique quadratic term. Thus, $\# \mu(K_4) > 1$. Since (7) is a representation of K_4 with two quadratic terms, it follows $\# \mu(K_4) = 2 = \lfloor \frac{4}{2} \rfloor$ and this theorem is valid.

Inductive hypothesis: Suppose this theorem is valid for all positive integers up to n .

Inductive step: Consider the graph K_{n+1} with vertices $\{v_0, v_1, \dots, v_n\}$.

Case 1 (n is even): By Lemma (I0), can write:

$$K_{n+1} = (v_n + v_{n-1}) \left(\sum_{i=0}^{n-1} v_i \right) + \mu(K_{n-2}). \quad (8)$$

By the inductive hypothesis, $\#\mu(K_{n-2}) = \lfloor \frac{n-2}{2} \rfloor = \frac{n-2}{2}$, since n is even. Thus, (8) is a representation for K_{n+1} with $\frac{n-2}{2} + 1 = \frac{n}{2} = \lfloor \frac{n+1}{2} \rfloor$. Such representation is minimum since K_{n+1} contains K_n as subgraph, which, by the inductive hypothesis, requires $\lfloor \frac{n}{2} \rfloor = \frac{n}{2}$ quadratic terms to be represented. Thus, $\#\mu(K_{n+1}) = \lfloor \frac{n+1}{2} \rfloor$.

Case 2 (n is odd): By Lemma (11), can write:

$$K_{n+1} = v_n \left(\sum_{i=0}^{n-1} v_i \right) + \mu(K_n). \quad (9)$$

By the inductive hypothesis, $\#\mu(K_n) = \lfloor \frac{n}{2} \rfloor = \frac{n-1}{2}$, since n is odd. Thus, (9) is a representation for K_{n+1} with $\frac{n-1}{2} + 1 = \frac{n+1}{2} = \lfloor \frac{n+1}{2} \rfloor$. Suppose this representation is not minimum. Then, there exists a representation of K_{n+1} that uses at most $\frac{n-1}{2}$ quadratic terms. By the construction of $\mu(K_n)$, $n \geq 3$, $n \bmod 2 = 1$, we have:

$$\mu(K_n) = \sum_{j=1}^{\frac{n-1}{2}} (v_{2j} + v_{2j-1}) \left(\sum_{i=0}^{2j-1} v_i \right). \quad (10)$$

By Theorem (5), the representation (10) is unique up to an isomorphism. Thus, if $\#\mu(K_{n+1}) = \#\mu(K_n)$ then in the minimum representation of K_{n+1} , v_n must be part of one or more of these quadratic terms to represent all the edges of K_{n+1} . Let μ_i be the i -th. term of $\mu(K_n)$ such that $\|\mu_i\| < \|\mu_{i+1}\|$, $1 \leq i \leq \frac{n-1}{2}$. Since in $\mu(K_n)$ the variables of each term μ_i belong to a factor of μ_{i+1} , it follows there exists a unique term containing v_{n-1} in one of the factors, namely, $\mu_{\frac{n-1}{2}}$. Hence, to represent the edge $\{v_{n-1}, v_n\}$, v_n should be inserted in another factor of $\mu_{\frac{n-1}{2}}$. There are two possibilities: repeat this process for all lower terms, or insert v_n in the two factors of $\mu_{\frac{n-1}{2}}$. If we always repeat the first process, we're not able to represent the edge $\{v_0, v_n\}$. If we choose to introduce v_n in the two factors of $\mu_{\frac{n-1}{2}}$, we'll have to get a way to represent an edge $\{v_{n-2}, v_n\}$ in the terms lower than $\mu_{\frac{n-1}{2}}$, since the vertices v_{n-2} and v_n form an independent set of vertices, according to Lemma (9). This alternative is not possible because v_{n-2} doesn't belong to any other term. Therefore, $\#\mu(K_{n+1}) > \#\mu(K_n)$. Since (9) is a representation for K_{n+1} of size $1 + \#\mu(K_n)$, it follows that $\#\mu(K_{n+1}) = 1 + \#\mu(K_n)$. By the inductive hypothesis, $\#\mu(K_n) = \lfloor \frac{n}{2} \rfloor$. Thus, $\#\mu(K_{n+1}) = 1 + \lfloor \frac{n}{2} \rfloor = 1 + \frac{n-1}{2} = \frac{n+1}{2} = \lfloor \frac{n+1}{2} \rfloor$ and the theorem follows, i.e., $\#\mu(K_n) = \lfloor \frac{n}{2} \rfloor$. \diamond

Next, we'll prove some lemmas and theorems which will be used to prove the exact value of $\#\mu(C_n)$.

Lemma 14. *If C_n is a circuit with n vertices, $n \bmod 2 = 0$, then C_n can be represented by:*

$$C_n = \sum_{i=0}^{\frac{n-4}{2}} (v_{0+i} + v_{n-i-2})(v_{n-i-1} + v_{0+i+1}).$$

Lemma 15. *If C_n is a circuit with n vertices, $n \bmod 2 = 1$, then C_n can be represented by:*

$$C_n = (v_{\frac{n-3}{2}} + v_{\frac{n-3}{2}+1})(v_{\frac{n-3}{2}+1} + v_{\frac{n-3}{2}+2}) + v_{\frac{n-3}{2}+1} + \sum_{i=0}^{\frac{n-5}{2}} (v_{0+i} + v_{n-i-2})(v_{n-i-1} + v_{0+i+1}).$$

Lemma 16. *If C_n is a circuit with n vertices, $n \bmod 2 = 0$, then C_n can be represented by:*

$$C_n = \sum_{i=0}^{\frac{n-4}{2}} (v_{0+i} + v_{n-i-2})(v_{n-i-1} + v_{0+i+1})$$

and such representation, using $\frac{n-2}{2}$ quadratic terms, is unique up to an isomorphism.

Lemma 17. *If C_n is a circuit with n vertices, $n \bmod 2 = 1$, then C_n can be represented by:*

$$C_n = (v_{\frac{n-3}{2}} + v_{\frac{n-3}{2}+1})(v_{\frac{n-3}{2}+1} + v_{\frac{n-3}{2}+2}) + v_{\frac{n-3}{2}+1} + \sum_{i=0}^{\frac{n-5}{2}} (v_{0+i} + v_{n-i-2})(v_{n-i-1} + v_{0+i+1})$$

and such representation, using $\frac{n-1}{2}$ quadratic terms, is unique up to an isomorphism.

Theorem 7. *If C_n is a circuit with n vertices, then C_n can be represented by:*

- a) $C_n = (v_{\frac{n-3}{2}} + v_{\frac{n-3}{2}+1})(v_{\frac{n-3}{2}+1} + v_{\frac{n-3}{2}+2}) + v_{\frac{n-3}{2}+1} + \sum_{i=0}^{\frac{n-5}{2}} (v_{0+i} + v_{n-i-2})(v_{n-i-1} + v_{0+i+1})$, if n is even,
- b) $C_n = \sum_{i=0}^{\frac{n-4}{2}} (v_{0+i} + v_{n-i-2})(v_{n-i-1} + v_{0+i+1})$, if n is odd.

Such representation is unique up to an isomorphism, with $\lfloor \frac{n-1}{2} \rfloor$ quadratic terms. **Proof** : It follows from Lemmas (I5) and (I4) that C_n can be represented in such form. In addition, it follows from Lemmas (I7) and (I6) that such representation is unique with this number of quadratic terms, up to an isomorphism. \diamond

Theorem 8. *If C_n is a circuit with n vertices then:*

$$\#\mu(C_n) = \lfloor \frac{n-1}{2} \rfloor.$$

Proof : From Theorem (I7) it follows that C_n can be represented uniquely up to an isomorphism, with $\lfloor \frac{n-1}{2} \rfloor$ quadratic terms. Therefore, $\#\mu(C_n) \leq \lfloor \frac{n-1}{2} \rfloor$.

Suppose $\#\mu(C_n) < \lfloor \frac{n-1}{2} \rfloor$ for some n . Let m the smallest value of n for which this happens. Then, m must be greater or equal to 5, since C_3 and C_4 are required to have at least one quadratic term to be represented and $\#\mu(C_3) = \#\mu(C_4) = 1$. Then, for any i such that $3 \leq i \leq m-3$, we have:

$$\begin{aligned} \#C_m^i &= \#C_i + \#C_{m-i+2} \\ &= \lfloor \frac{i-1}{2} \rfloor + \lfloor \frac{m-i+1}{2} \rfloor. \end{aligned}$$

Case 1 (m and i are odd) :

$$\#C_m^i = \frac{i-1}{2} + \frac{m-i}{2} = \frac{m-1}{2} = \lfloor \frac{m-1}{2} \rfloor.$$

Case 2 (m and i are even) :

$$\#C_m^i = \frac{i-2}{2} + \frac{m-i}{2} = \frac{m-2}{2} = \lfloor \frac{m-1}{2} \rfloor.$$

Case 3 (m is even and i is odd) :

$$\#C_m^i = \frac{i-1}{2} + \frac{m-i+1}{2} = \frac{m}{2} > \lfloor \frac{m-1}{2} \rfloor.$$

Case 4 (m is odd and i is even) :

$$\#C_m^i = \frac{i-2}{2} + \frac{m-i+1}{2} = \frac{m-1}{2} = \lfloor \frac{m-1}{2} \rfloor.$$

Thus, $\#\mu(C_m) = \min\{\#C_m^i : 3 \leq i \leq m-3\} = \lfloor \frac{m-1}{2} \rfloor$ and there is no such m . Therefore,

$$\#\mu(C_n) = \lfloor \frac{n-1}{2} \rfloor. \diamond$$

Theorem 9. *Let K_n^1 be the graph $K_n \setminus \{a\}$, where a is any edge of $E(K_n)$. Then $\#\mu(K_n^1) \leq \lfloor \frac{n-1}{2} \rfloor$.*

Proof : Without loss of generalization, suppose $a = \{v_{n-1}, v_{n-2}\}$. If n is odd then K_n^1 can be represented by:

$$K_n^1 = (v_{n-1} + v_{n-2}) \left(\sum_{i=0}^{n-3} v_i \right) + \sum_{j=1}^{\frac{n-3}{2}} ((v_{2j} + v_{2j-1}) \left(\sum_{i=0}^{2j-1} v_i \right) + v_{2j-1}),$$

since by Lemma (10), K_n can be represented by:

$$K_n = \sum_{j=1}^{\frac{n-1}{2}} ((v_{2j} + v_{2j-1}) \left(\sum_{i=0}^{2j-1} v_i \right) + v_{2j-1}),$$

with $\frac{n-1}{2}$ quadratic terms, i.e.,:

$$\# \mu(K_n^1) \leq \frac{n-1}{2} = \lfloor \frac{n-1}{2} \rfloor.$$

If n is even then K_n^1 can be represented by:

$$K_n^1 = (v_{n-1} + v_{n-2} + v_{n-3}) \left(\sum_{i=0}^{n-3} v_i \right) + v_{n-3} + \sum_{j=1}^{\frac{n-4}{2}} ((v_{2j} + v_{2j-1}) \left(\sum_{i=0}^{2j-1} v_i \right) + v_{2j-1}),$$

since by Lemma (11), K_n can be represented by:

$$K_n = v_{n-1} \left(\sum_{i=0}^{n-2} v_i \right) + \sum_{j=1}^{\frac{n-2}{2}} ((v_{2j} + v_{2j-1}) \left(\sum_{i=0}^{2j-1} v_i \right) + v_{2j-1}),$$

with $\frac{n-2}{2}$ quadratic terms, i.e.,:

$$\# \mu(K_n^1) \leq \frac{n-2}{2} = \lfloor \frac{n-1}{2} \rfloor.$$

Thus, the theorem follows. \diamond

9 Bounds for the Structure

Theorem 10. Let $f \in F^c[n; \leq 2]$ and let G_f be the element of G^n associated to f . Then $\mu(f)$ is a minimum form of f iff $\mu_f = \# \mu(f)$, i.e., the number of quadratic terms in $\mu(f)$ is the minimum number of complete bipartite or tripartite subgraphs to cover G_f .

Proof : (\implies) Let $\mu(f)$ be a minimum form of f . Then, the quadratic terms of f are, by definition, pure or not pure. By Lemmas (8) and (9), we know that the pure terms are associated to bipartite complete graphs and the terms which aren't pure, complete tripartite graphs. Since the isomorphism of Theorem (1) $\Phi = (\varphi_1, \varphi_2)$ associates to each variable x_i of f the vertex v_i of G_f , it follows that such bipartite and tripartite graphs are subgraphs of G_f .

(\impliedby) This direction is trivial. \diamond

Theorem 11. *Let G be a graph of order n and let $\beta(G)$ be a minimum cover of G by bipartite subgraphs. Then, if $\mu(G)$ is a minimum cover of G with μ_2 bipartite subgraphs and μ_3 tripartite subgraphs, we have:*

$$\mu_2 + \mu_3 \leq \#\beta(G) \leq 2\mu_3 + \mu_2.$$

10 SERPENT

SERPENT is a former AES candidate. Applying the integer vector coding, as explained in Section 4, and the bounds proved in Sections 7-9, we found 6 linearly independent relations with deviation $\delta = 1/2$ (i.e., maximum deviation). They are:

$$x_2 = y_1y_0 + y_3 + y_2 + y_1 + y_0$$

$$x_3x_2 + x_3x_1 + x_3x_0 + x_2x_0 + x_1x_0 + x_3 + x_2 = y_2y_1 + y_2y_0 + y_3 + y_2 + y_1$$

$$x_3x_1 + x_3x_0 + x_1x_0 + x_1 + x_0 = y_3 + y_1 + y_0$$

$$x_3x_1 + x_1x_0 + x_3 + x_2 = y_1 + y_0$$

$$x_3x_1 + x_2x_0 + x_3 + x_2 = y_3y_1 + y_2y_1 + y_2y_0 + y_1y_0 + y_2 + y_1$$

$$x_3 + x_0 = y_3y_1 + y_3y_0 + y_1y_0 + y_3 + y_2 + y_0$$

Using these relations and considering SERPENT reduced to one round, we explain in a forthcoming paper [8] that the linear transformation in the round is weak; more specifically, we show the linear transformation used in each iteration of SERPENT is weak for 75% of the involved bits. Furthermore, we give indications on how to find differential and linear approximations with high probabilities.

11 CRYPTON

CRYPTON is also a former AES candidate. Applying the integer vector coding, as explained in Section 4, and the bounds proved in Sections 7-9, we also found 4 linearly independent relations with deviation $\delta = 1/2$. Actually, we are able to combine linear relations and get many quadratic relations for the 8×8 S-boxes – and obtained 16 such relations for each S-box, with deviation $\frac{1}{2}$ (i.e., maximum deviation). Using these relations we describe in a forthcoming paper [8] how to partially break one round of CRYPTON.

12 Conclusions

We introduce polynomial approximations and consider the particular case of quadratic approximations. We establish an isomorphism between the set of quadratic Boolean functions and graphs. We present the problem of finding a minimum representation of quadratic functions, prove bounds on the number of terms and variables. We found quadratic relations for SERPENT and CRYPTON with the highest probabilities.

References

1. E. Biham and A. Shamir, "Differential cryptanalysis of the full 16-round DES", Proc. of CRYPTO'92, Lec. Notes in C.S, Springer-Verlag, 1992
2. E. Biham and A. Shamir, "Differential cryptanalysis of Data Encryption Standard", Springer-Verlag, 1993.
3. M. Matsui, "The first experimental cryptanalysis of the Data Encryption Standard", Proceedings of Crypto'94, Lec. Notes in C.S. number 839, Springer-Verlag, 1994
4. National Bureau of Standards, "Data Encryption Standards", FIPS Publication 46, U. S. Dept. of Commerce, 1977
5. R. Anderson, E. Biham and L. Knudsen, "Serpent: a proposal for the Advanced Encryption Standard", AES proposal available on:
<http://csrc.nist.gov/encryption/aes/>
6. C. H. Lim, "CRYPTON: A new 128-bit block cipher", AES proposal available on:
<http://csrc.nist.gov/encryption/aes/>
7. C. H. Lim, "A revisited version of CRYPTON: CRYPTON V1.0", Proceedings of Fast Software Encryption 1999, Lec. Notes in C.S. Springer-Verlag, 1999
8. P.G. Pinheiro and R. Terada, "Quadratic cryptanalysis of SAFER and CRYPTON" – working paper (May 2001).
9. Thomas Jakobsen, "Cryptanalysis of Block Ciphers with Probabilistic Non-Linear Relations of Low Degree". Crypto'98, in Lecture Notes in Computer Science, Springer-Verlag, 1998.
10. Shiho Moriai, Takeshi Shimoyama, Toshinobu Kaneko, "Interpolation Attacks of the Block Cipher: SNAKE", Fast Software Encryption Workshop'99, FSE'99, (Rome, March, 1999).

Approximate Power Roots in \mathbb{Z}_m^*

Ismael Jiménez Calvo¹ and German Sáez Moreno²

¹ C.S.I.C, Instituto de Física Aplicada, c/Serrano 144, 28006-Madrid, Spain.

`ltqij04@pinar2.csic.es`

² Dep. de Matemàtica Aplicada IV, Universitat Politècnica de Catalunya, c/Jordi Girona, 1-3, 08034-Barcelona, Spain.

`german@mat.upc.es`

Abstract. Some algorithms to search for power residues close to a desired value are reported. They are used in the cryptanalysis of low exponent RSA with partially known clear text.

1 Introduction

We define the approximate power roots in \mathbb{Z}_m of ρ as the solutions to the congruence $x^n \equiv \rho + \varepsilon \pmod{m}$, where ε takes low absolute values.

In [12], the set of elements whose squares mod n are small enough using lattices are studied. A description of the gaps between such elements is obtained, and two algorithms that find elements with small modular squares are obtained. The problem of finding small modular squares when the modulus is a large composite number of unknown factorization is a well known technique for factoring the modulus by the use of such congruences of squares. It is interesting to point out that this problem is the particular case, of $n = 2$ and $\rho = 0$, of the more general problem here treated.

The review on cryptanalysis [2], p. 584, presents some specific methods for the approximation of quadratic and cubic residues of interest in the evaluation of the security of some signature schemes. In addition to these cryptologic problems, the main purpose of this work is centered on the cryptanalysis of a RSA cryptosystem [10] with a low public exponent, and whenever the clear message is partially known. Attacks to RSA in this scenario have been studied by H. Hastad [5] and D. Coppersmith [4] using the LLL algorithms [6] as a tool for finding small integer solutions to polynomials modulo m . Actually, Coppersmith presents a procedure for deciphering the clear message when it is encrypted with exponent 3 and the cryptanalyzer already knows two-thirds of it. The methods in this paper do not use any base reduction algorithm such as the LLL ones. They are based instead on an approach that consists of studying the distribution of the power residues for values of x near a simple fraction of the modulus.

Fig. 1 shows a typical distribution of quadratic residues. When integers are close to a simple fraction a/b of the modulus m , their quadratic residues appear

* The work of the second author was partially supported by *Ministerio de Ciencia y Tecnología* under project TIC 2000-1044

to be distributed in a set of parabolas. The number of parabolas coincides with the denominator b when it is odd, or with $b/2$ when even. The vertices of the parabolas are at multiples of m/b^2 which are equally spaced. These observations are generalized to higher power residues and analyzed in Section 2. The results for the distribution of power residues as integer points of polynomials are exploited to search for approximate power residues in Section 3. The special case of quadratic residues and an alternative approach to the Montgomery polynomials used in the MPQS factoring algorithm [8], which can be generalized here to higher powers, are also analyzed. Section 4 deals with the application of these techniques to the cryptanalysis of low public exponent RSA, and results of the practical implementation of the algorithms are presented. The best bound for ε in power root approximation is of $O(m^{(n-1)/n})$.

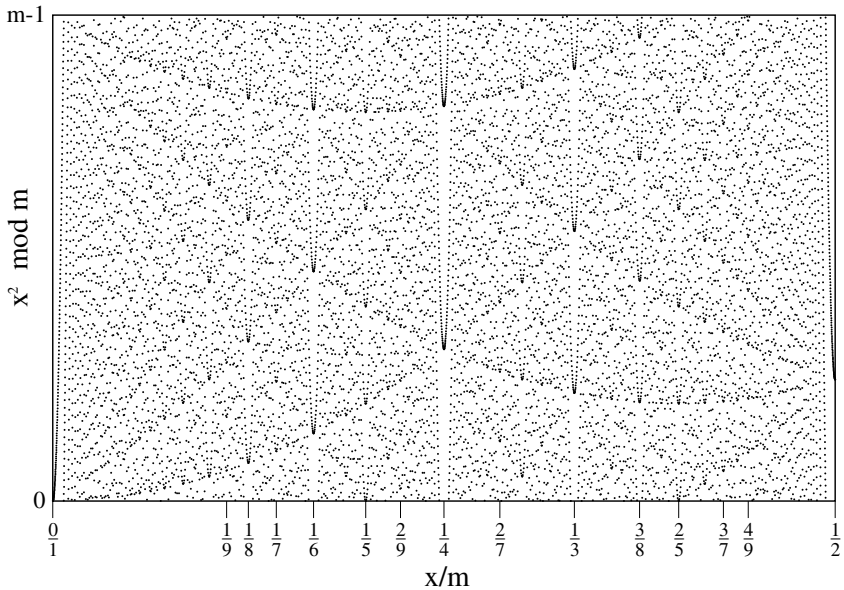


Fig. 1. Quadratic residues

2 Power Residues Near a Modulus Fraction

Power residues in \mathbb{Z}_m near simple fractions of the modulus exhibit geometrical patterns that are sometimes easy to recognize with proper graphical representations. In this Section, the mathematical grounds for those facts are developed.

Let two mutually prime integers be a and b and a modulus m , and let

$$x_0 = \left\lfloor \frac{a}{b} m \right\rfloor.$$

where $\lfloor x \rfloor$ stands for the integer nearest to x .

Proposition 1. *The n^{th} power residue of $x_0 \bmod m$ is the integer nearest to some multiple of the rational m/b^n .*

Proof. We define

$$\alpha \equiv am \pmod{b}, \quad -b/2 < \alpha \leq b/2,$$

so, $x_0 = \frac{am-\alpha}{b}$. We have,

$$\begin{aligned} x_0^n &= \left(\frac{am-\alpha}{b} \right)^n = \frac{1}{b^n} \sum_{k=0}^n \binom{n}{k} a^k m^k (-\alpha)^{n-k} = \\ &= \frac{1}{b^n} m \sum_{k=1}^n \binom{n}{k} (-1)^{n-k} a^k m^{k-1} \alpha^{n-k} + (-1)^n \left(\frac{\alpha}{b} \right)^n \end{aligned}$$

If we define the integer β_n as,

$$\beta_n \equiv \sum_{k=1}^n \binom{n}{k} (-1)^{n-k} a^k m^{k-1} \alpha^{n-k} \pmod{b^n}, \quad 0 \leq \beta_n < b^n, \quad (1)$$

we can write,

$$x_0^n = \frac{m}{b^n} \left(\sum_{k=1}^n \binom{n}{k} (-1)^{n-k} a^k m^{k-1} \alpha^{n-k} - \beta_n \right) + \beta_n \frac{m}{b^n} + (-1)^n \left(\frac{\alpha}{b} \right)^n.$$

We say r_n to the residue of $x_0^n \pmod{m}$. By (1), the term in brackets is divisible by b^n , and reducing modulo m we finally have,

$$r_n \equiv \beta_n \frac{m}{b^n} + (-1)^n \left(\frac{\alpha}{b} \right)^n \pmod{m}. \quad (2)$$

Since $-b/2 < i \leq b/2$ then $|\alpha/b|^n \leq 1/2^n$ and may be rejected by rounding the first term up or down to the nearest integer. \square

Remark 1. The calculation of β_n can be performed by the operation,

$$\beta_n \equiv \left\lfloor \frac{r_n b^n}{m} \right\rfloor \pmod{m}. \quad (3)$$

Nevertheless, as is implicit in (2), and taking into account that b^n divides $\beta_n m + (-1)^n \alpha^n$, or as easily follows from (1) using the definition of α , the expression,

$$\beta_n \equiv (-1)^{n+1} \alpha^n m^{-1} \pmod{b^n} \quad (4)$$

may be used with advantage because it operates in $\mathbb{Z}_{b^n}^*$, a group smaller than \mathbb{Z}_m^* in the practical cases.

Remark 2. Reducing equation (4) modulo b , and taking into account that $\alpha \equiv ma \pmod{b}$,

$$\beta_n \equiv (-1)^{n+1} a^n m^{n-1} \pmod{b}. \quad (5)$$

From which it follows that β_n modulo b is never zero whenever $\gcd(b, m) = 1$. In fact, if $a \equiv \pm(-1)^{n+1} m^{(n-1)/n} \pmod{b}$, then $\beta_n \equiv \pm 1 \pmod{b}$. This fact will be used later in the generalization of the Montgomery polynomials which yield power residues with low absolute values. Note also that $\beta_1 = a$.

We now analyze the n^{th} power residues near x_0 , say $(x_0 + i)^n$, for a low absolute value of i .

Proposition 2. *The n^{th} power residues near x_0 are separated approximately by a multiple of $h = m/b'$, where $b' = b^{n-1}/\gcd(n, b^{n-1})$.*

Proof. Let $R_i \equiv (x_0 + i)^n \pmod{m}$. Expanding the binomial and reducing modulo m , we can write,

$$R_i \equiv r_n + \binom{n}{1} r_{n-1} i + \binom{n}{2} r_{n-2} i^2 + \dots + \binom{n}{n-1} r_1 i^{n-1} + \binom{n}{n} i^n \pmod{m}.$$

Taking into account (2),

$$\begin{aligned} R_i &\equiv r_n + \binom{n}{1} \beta_{n-1} \frac{m}{b^{n-1}} i + \binom{n}{2} \beta_{n-2} \frac{m}{b^{n-2}} i^2 + \dots + \binom{n}{n-1} \beta_1 \frac{m}{b} i^{n-1} \\ &\quad - \left(-\frac{\alpha}{b}\right)^n + \binom{n}{0} \left(-\frac{\alpha}{b}\right)^n + \binom{n}{1} \left(-\frac{\alpha}{b}\right)^{n-1} i + \dots + \binom{n}{n} i^n \pmod{m}. \end{aligned}$$

Then,

$$R_i \equiv r_n + \sum_{k=1}^{n-1} \binom{n}{k} \beta_{n-k} \frac{m}{b^{n-k}} i^k + \left(i - \frac{\alpha}{b}\right)^n - \left(-\frac{\alpha}{b}\right)^n \pmod{m}.$$

We define $\delta = \left(i - \frac{\alpha}{b}\right)^n - \left(-\frac{\alpha}{b}\right)^n$, which is low as far as low i values are concerned.

$$R_i - r_n - \delta = \sum_{k=1}^{n-1} \binom{n}{k} \beta_{n-k} \frac{m}{b^{n-k}} i^k - \ell m,$$

for some integer ℓ . If we divide by $h = m/b'$ and we denote $d = \gcd(n, b^{n-1})$,

$$\frac{R_i - r_n - \delta}{h} = \sum_{k=1}^{n-1} \frac{1}{d} \binom{n}{k} \beta_{n-k} b^{k-1} i^k - \ell b'.$$

It is a matter of checking that d divides the terms $\binom{n}{k} b^{k-1}$ in the summation; then it is an integer. Finally, We can write,

$$\frac{R_i - r_n - \delta}{h} \equiv \frac{1}{d} \sum_{k=1}^{n-1} \binom{n}{k} \beta_{n-k} b^{k-1} i^k \pmod{b'}, \quad (6)$$

since h divides $R_i - r_n - \delta$ exactly, the statement of the proposition holds for low i values. \square

The modular equation (6) ensures that the residues are approximately repeated, at least at intervals of i of every b' . It then becomes natural to consider the power residues near x_0 in relation to the index i and the variable y in the following way,

$$x = x_0 + i + b'y, \quad -b'/2 < i \leq b'/2, \quad y \in \mathbb{Z}_m.$$

We define the set of b' polynomials indexed by i ,

$$P_i(y) = (x_0 + i + b'y)^n - lm,$$

for some integer l depending on x_0 , i and y , such that $0 \leq P_i(y) < m$. The polynomials have unique extreme points, we say vertex if n is even, or inflection point if it is odd.

Proposition 3. *Every polynomial $P_i(y)$ has the unique vertex (inflection point if n is odd) for $y_v = -\frac{1}{b'}(i - \frac{\alpha}{b})$ and $P_i(y_v) = \beta_n \frac{m}{b^n} + K \frac{m}{b'}$ for some K in $\mathbb{Z}_{b'}$.*

Proof. Let

$$K \equiv \frac{1}{d} \sum_{k=1}^{n-1} \binom{n}{k} \beta_{n-k} b^{k-1} i^k \pmod{b'}. \quad (7)$$

Note that if i is substituted by $i + b'y$, K remains unchanged. Let us now consider the value of $R_{i+b'y} = P(y)$ given by (6). Substituting r_n by its value given in (2), and taking into account that, in this case, $\delta = (i + b'y - \frac{\alpha}{b})^n - (-\frac{\alpha}{b})^n$, finally we can write,

$$P_i(y) = \beta_n \frac{m}{b^n} + K \frac{m}{b'} + (i + b'y - \frac{\alpha}{b})^n. \quad (8)$$

Note that the derivatives of $P_i(y)$ vanish for y_v which concludes the proof. \square

Remark 3. The polynomials $P_i(y)$ are the polynomial $P(Y) = b'^n Y^n$ translated by the vectors $\mathbf{u} = (y_v, P_i(y_v))$.

Remark 4. The first power residue of a polynomial $P_i(y)$, (i.e. for $y = 0$), is at $(i - \frac{\alpha}{b})^n$ from the vertex.

Remark 5. Approximation of power residues is performed by selecting the appropriate K value, and subsequently the solution to equation (7), to obtain the index i of the polynomial we are looking for. The solution is treated in the Appendix, where it is shown that it exists for any K value in $\mathbb{Z}_{b'}$, except when n is even and greater than 2, and, at the same time, $b = 2t$ with t being odd. In this case, only even K values are allowed. Thus, the power residues near a fraction of the modulus are distributed in b' polynomials ($b'/2$ if the above exception condition holds), equally spaced by h or by $2h$ if the above special condition holds (see Fig. 2).

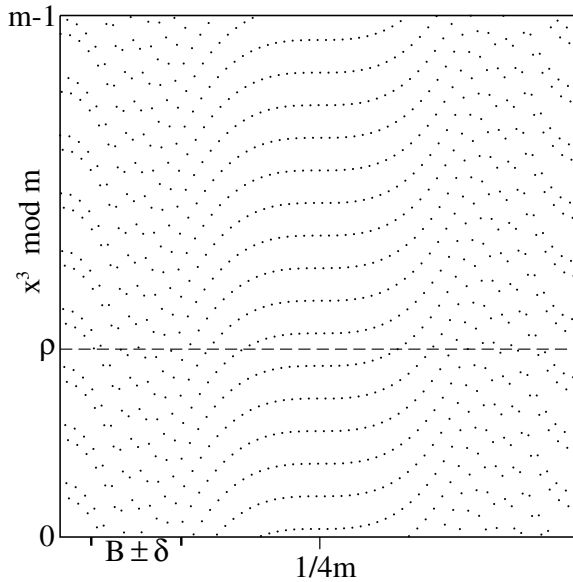


Fig. 2. Cubic residues near $1/4m$. $a = 1$, $b = 4$, $b' = 16$.

3 Approximation of Power Roots

Suppose we want to find an approximation x to an n^{th} power residue ρ for close to $x_0 = \lfloor \frac{a}{b}m \rfloor$. In other words, we seek a solution of $x^n \equiv \rho + \varepsilon \pmod{m}$, where ε takes low absolute values and close to x_0 . We use the notation and results introduced in Section 2, choosing the polynomial whose vertex is the nearest to ρ (see fig. 3) if n is odd. Because the two stems of the polynomial are upwards whenever n is even, the chosen polynomial must have its vertex close to and below ρ . Wherever all possible values for K in $\mathbb{Z}_{b'}$ are possible (see Remark 5 of Proposition 3), these conditions are achieved when $0 \leq |P_i(y_v) - \rho| < h/2$ for n odd, and $0 \leq P_i(y_v) - \rho < h$ for n even, with $h = m/b'$. Proposition 3 shows that $P_i(y_v) - \rho = \beta_n \frac{m}{b^n} + Kh - \rho$, with K given by (7). Dividing the inequalities by h , we can verify that

$$K = \begin{cases} \left\lfloor \frac{\rho}{h} - \frac{\beta_n}{db} \right\rfloor & \text{for } n \text{ odd,} \\ \left\lfloor \frac{\rho}{h} - \frac{\beta_n}{db} \right\rfloor & \text{for } n \text{ even.} \end{cases} \quad (9)$$

Substituting the value of K in (7) and following the procedure explained in the Appendix, the index i can be calculated. Whenever $n = 2s$, $s \geq 2$ and $b = 2t$, t being odd, only even values of K are possible, and, if an odd value for K is obtained in (9), the closest even value under it must be used for the calculus of

i . In this case, two values of i are solutions of (7). Once the index i is known, it is only a matter of finding the integer y whose image is the nearest to ρ . Let us consider the polynomial $P(Y) = b'^n Y^n$, which is the one carried from $P_i(y)$ by the vector $\mathbf{u} = (y_v, P_i(y_v))$ (see Remark 3 in Proposition 3). Let

$$t = \rho - P_i(y_v) = \rho - (\beta_n \frac{m}{b^n} + Kh), \quad (10)$$

then, $t = b'^n Y^n$ gives the ordinate Y when the polynomial crosses ρ . Since $y = Y + y_v = Y - \frac{1}{b'} (i - \frac{\alpha}{b})$, we have

$$y = \begin{cases} \frac{t^{\frac{1}{n}} - (i - \frac{\alpha}{b})}{b'} & \text{for } n \text{ odd,} \\ \frac{\pm t^{\frac{1}{n}} - (i - \frac{\alpha}{b})}{b'} & \text{for } n \text{ even.} \end{cases} \quad (11)$$

The nearest integer to y gives $x = x_0 + i + b'y$ as an approximate power root of ρ near x_0 .

It must be noted that this procedure only works when the first integer point of any polynomial $P_i(y)$ is at most at $h/2$ from the vertex, i.e. $\pm h/2 = (i - \alpha/b)^n$ (see Remark 4 in Proposition 3). In a first approximation, we can consider the greatest absolute value of $i = b'/2$ and reject the term α/b . With this assumption, we have that $b < 2^{1/(n+1)} d^{1/(n-1)} m^{1/(n^2-1)} \approx m^{1/(n^2-1)}$. The greatest error in the approximate power root can be estimated for n even by,

$$\varepsilon = \frac{P(Y+1) - P(Y)}{2}, \quad \text{for } P(Y) = b'^n Y^n = \frac{m}{b'}. \quad (12)$$

Using the approximation $P(Y+1) - P(Y) \approx b'nY^{n-1}$, we can obtain

$$\varepsilon \approx \frac{n}{2d^{\frac{1}{n}}} (bm)^{\frac{n-1}{n}}; \quad (13)$$

that is, the approximation is $O((bm)^{(n-1)/n})$.

Whenever the numerator a can be selected freely for a fixed denominator b , an improvement can be achieved. We wish to search for a polynomial with the vertex close to ρ . In other words, we wish to minimize the parameter t defined in (10). Let $\beta^* = \lfloor \rho b^n / m \rfloor$. Then

$$t \approx \beta^* \frac{m}{b^n} - \beta_n \frac{m}{b^n} - Kh = \frac{(\beta^* - \beta_n)}{bd} h - Kh.$$

If $\beta^* - \beta_n \equiv \delta \pmod{bd}$ with $\delta \in \{0, -1, +1\}$, an a in equation (7) can be found such that t is bounded by m/b^n . When $d = 1$, from (5) in Remark 2 of Proposition 1, it easily follows that

$$a \equiv ((-1)^{n+1} (\delta - \beta^*) m^{1-n})^{\frac{1}{n}} \pmod{b}. \quad (14)$$

The greatest errors are calculated as in (12). In this case, the vertex of the polynomial nearest to the residue to be approximated is at most at $2m/b^n$ from

the residue. This leads to an error bound of $O(m^{(n-1)/n})$, which only holds if the polynomial has at least one integer point at the interval of $2m/b^n$ from the vertex. That occurs only when

$$b < 2^{\frac{n+1}{n^2}} d^{\frac{1}{n}} m^{\frac{1}{n^2}}, \quad (15)$$

as can be calculated by the same procedure described above, when the numerator a cannot be freely chosen.

Note that a better bound in the approximation of the power residue is achieved. However, the power root cannot be approximated. But even in this case, an approximation of the power root to a desired value can be performed by the following procedure. A value $b = b_1 b_2 \cdots b_k$ is selected as the product of the lowest primes that do not divide n (this assures that $d = 1$), and for which (14) modulo b_i has the greatest number of solutions. Let q_i^j be the solutions, where i runs from 1 to k and j runs over the number of solutions for each prime b_i , i.e. from 1 to $J_i = \gcd(n, b_i - 1)$ (see, for example, Corollary 2.42 of Theorem 2.41 in [9], p. 104). We follow the method of the Chinese Remainder Theorem to calculate the set of values for a by means of

$$a \equiv \sum_{i=1}^k e_i q_i^j \pmod{b}, \quad j \in \{1, 2, \dots, J_i\}.$$

Here $e_i = \prod_{i \neq j} b_j b_j^{-1}$, b_j^{-1} being the inverse of b_j modulo b_i . A proper combinatorial optimization method can help us to find a solution for a , near to the desired value.

3.1 Generalization of Montgomery Polynomials

P. Montgomery suggested a way for choosing polynomials that supply low quadratic residues used in the MPQS to factor large integers [8]. A variation of this method was described in the HMPQS of R. Peralta [7]. Here, an alternative method to search for these types of polynomials and a generalization to higher powers residues is presented.

The method is the same as that explained in the last Section, when the numerator a can be selected freely, applied to the case of *approximation to zero power roots*, i.e. the case of $\beta^* \equiv 0 \pmod{b}$. It must be noted from (14) that δ can never be zero whenever $\gcd(m, b) = 1$, then $\delta = \pm 1$ for n odd and $n = -1$ for n even. For n even the two stems of the polynomials can yield low power residues, while for n , the left stem must be chosen if $\delta = +1$ or the right stem if $\delta = -1$. The procedure is as follows; $b = O(m^{1/n^2})$ is selected such that m is a n^{th} root mod b and

$$a \equiv (-1)^n m^{\frac{1-n}{n}} \pmod{b}, \quad (16)$$

for n even, otherwise $-a$ is also a valid choice. As occurs in (13), it can be observed that the size of residues are in inverse proportion to $d^{1/n}$. Then, when $d \neq 1$, smaller residues are obtained, but a must fulfill the above equation modulo

b , and also modulo $d \gcd(b, d)$, which is quite probable because d is usually small. The calculation of x_0 and b' is performed as usual. Finally, i is calculated by the procedure described in the Appendix, prior to the calculation of K by (9) with $\rho = 0$, leading to

$$P_i(y) = (x_0 + i + b'y)^n$$

as the polynomial we are looking for.

3.2 Approximation of Quadratic Residues

The number of parabolas b' containing the quadratic residues near a fraction is b in this case, for b odd or $b/2$ if even. An example is depicted in Fig. 3. Taking into account that $\beta_1 = a$ (see Remark 2 in Proposition 1), the equation (7) takes the form

$$K \equiv \frac{2a}{d}i \pmod{b'}.$$

The i index of the parabola containing the quadratic residue closest to ρ (P_2 in the example) is calculated now in a straightforward way, substituting the value of K from (9),

$$i \equiv \left(\frac{2a}{d}\right)^{-1} \left\lfloor \frac{r_2 - \rho}{h} \right\rfloor \pmod{b'}, \quad b' = \begin{cases} b, & \text{if } b \text{ odd} \\ b/2, & \text{if } b \text{ even} \end{cases}, \quad h = \frac{m}{b'}, \quad (17)$$

being $d = 1$ or 2 according to the parity of b . Finally, the value of y in (11) for the nearest residue of the parabola P_i to ρ takes the expression

$$y = \left\lfloor \frac{i}{b'} \pm \frac{\sqrt{i^2 + \rho - r_i}}{b'} \right\rfloor,$$

giving $x_0 + i + b'y$ as an approximate square root of $\rho \bmod m$. The greatest error ε with this method is $\sqrt{b'm}$, which follows from (13).

When we are free to seek the approximate square root along \mathbb{Z}_m , we can choose the numerator a by means of (14), which leads to errors of $\varepsilon \approx \sqrt{m}$. The selection of an a near a desired value can be performed following the method for residues of higher order explained above, selecting a highly composite denominator b . So, when we search for low quadratic residues, we select a $b \approx (8m)^{1/4}$ (this ensures that the residues are inside the $\pm m/b^2$ interval), which factors as $b_1 b_2 \dots b_k$, b_i being the lowest distinct odd primes such that m is a quadratic residue modulo b_i . In this way, equation (16) has 2^k solutions to choose the value most fitted to an ideal a . We compute first

$$a \equiv m^{-1/2} \equiv \pm q_i \pmod{b_i}, \quad i = 1 \dots k.$$

The Chinese Remainder Theorem gives the possible solutions for a :

$$a \equiv \sum_{i=1}^k t_i e_i q_i \pmod{b}, \quad t_i \in \{+1, -1\},$$

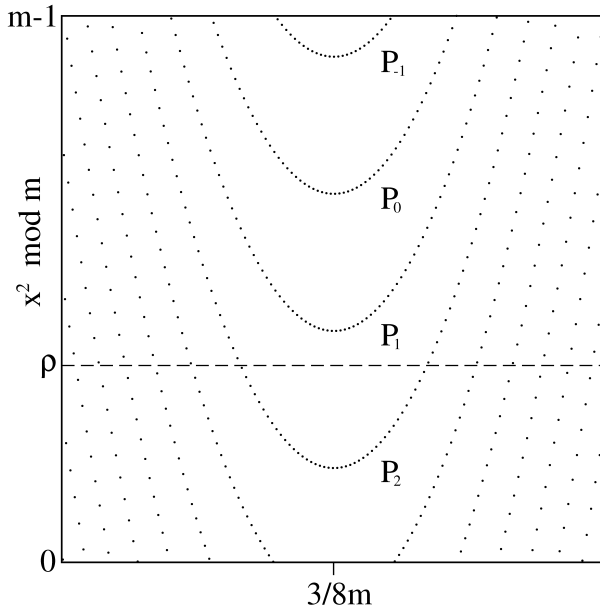


Fig. 3. Quadratic residues near $3/8m$. $a = 3$, $b = 8$, $b' = 4$.

where $e_i = \prod_{j \neq i} b_j b_j^{-1}$, b_j^{-1} being the inverse of b_j modulo b_i . A combinatorial optimization method can select the t_i coefficients that lead to the fittest a value. Although the approximation in the quadratic residue is bounded by \sqrt{m} , the approximation in the square root is worse and less predictable than in the previously described methods. If k is the number of factors of b , the square root $x = x_0 + \delta$, has a maximum indetermination always $\delta > m/2^k$. Typical values are $k = 16$ for $m \approx 10^{100}$ and $k = 27$ for $m \approx 10^{200}$.

4 Application to Low Exponent RSA Cryptanalysis

Suppose that a clear text C , of which the cryptanalizer knows a part of its most significant bits, has been encrypted using a RSA scheme [10] with a low public exponent n and a modulus m . Let ρ be the cipher text, then

$$\rho \equiv C^n \pmod{m}.$$

The procedure to recover the entire clear text without knowing the private exponent can be visualized in Fig. 2. Let $C = B \pm \delta$, for an unknown δ , which is near to some fraction $\frac{a}{b}$ of the modulus. Selecting the set of polynomials that cross the ordinate ρ for the abscissa interval $B \pm \delta$, we can determine which of them perform the procedure for an integral δ that permits the recovery of the clear text. We use Proposition 3, and especially Remark 3. Each polynomial $P_i(y)$, which has its integer points as the n^{th} power residues of $x = x_0 + i + b'y$, is

the polynomial $P(Y) = b'^n Y^n$ carried over according to the following coordinate change

$$Y = y + \frac{1}{b'}(i - \frac{\alpha}{b}); \quad P(Y) = P_i(y) - \left(\beta_n \frac{m}{b^n} + K \frac{m}{b'}\right),$$

with K depending on i as in (7). It follows that for $x = B$,

$$b'Y = B - x_0 - \frac{\alpha}{b} \quad \text{and then,} \quad \rho - \left(\beta_n \frac{m}{b^n} + K \frac{m}{b'}\right) = (B - x_0 - \frac{\alpha}{b})^n,$$

which enables us to determine K as,

$$K = \begin{cases} \left\lfloor \frac{\rho b^n - \beta_n m - (bB - bx_0 - \alpha)^n}{mbd} \right\rfloor & \text{for } n \text{ odd,} \\ \left\lfloor \frac{\rho b^n - \beta_n m - (bB - bx_0 - \alpha)^n}{mbd} \right\rfloor & \text{for } n \text{ even.} \end{cases} \quad (18)$$

The procedure in the Appendix enables us to recover the index i from (7), which defines the polynomial that may contain ρ as an integer point. It is a matter of calculating the variable y at which the image of the polynomial is ρ and to check whether it is an integer. Note that we use the value of K , not the reduced modulo b' , because we consider that the polynomials are not truncated as they are in Fig. 2. Because

$$P(Y) = (b'Y)^n \equiv \rho - \beta_n \frac{m}{b^n} - K \frac{m}{b'} \pmod{m} \quad \text{and} \quad Y = y + \frac{1}{b'}(i - \frac{\alpha}{b}),$$

we obtain that

$$y = \frac{(\rho b^n - \beta_n m - K m d b)^{\frac{1}{n}} - b i + \alpha}{b b'}.$$

If y is an integer, then $C = x_0 + i + b'y$. Otherwise we calculate y for $K \pm 1, K \pm 2 \dots$ until it is successfully achieved. It must be noted that $t = \rho b^n - \beta_n m - K m d b$ must be a perfect n^{th} power because, otherwise y is irrational, a condition that can be very quickly tested. In the case of n even, only positive values of t must be used and the positive and negative roots must be checked.

5 Conclusions

The geometrical order exhibited by power residues near simple fractions of the modulus has been exploited to search for approximate power residues. When we are obliged to seek an n^{th} power residue near some fixed fraction $\frac{a}{b}m$, the achieved approximation is of the order of $(bm)^{(n-1)/n}$ and the denominator b of the fraction must be under $m^{1/(n^2-1)}$. If the desired power root lies along 0 to $m-1$, i.e. the numerator a may be chosen freely, the approximation of the residue is of the order of $m^{(n-1)/n}$, while b must be under m^{1/n^2} . The procedures developed in this paper do not depend on the arithmetic structure of the modulus. The results are applied to search for low quadratic residues, in a way equivalent, but not equal, to the *Quadratic Sieve* factoring algorithm. Low public exponent

RSA cryptanalysis can also be broached with these methods. It is known that smart credit cards use low public RSA exponents, and these methods provide means to perform attacks when the most significant bits of the clear text are known. Nevertheless, the general case only works for exponent $n = 2$. In this case, knowing $2/3$ of the bits of the clear text, the remaining $1/3$ of the bits can be recovered. Tests of the algorithm explained in Section 4, programmed using the *Pari Package* [1] in a Pentium processor at 400 Mhz, take only few milliseconds to carry out this task with numbers of 1024 bits. It easily follows from the theory of Farey sequences (see, for example, Theorem 6.7 in [9]) that the separation between contiguous Farey fractions of order b are under $\Delta = m/b^2$, while the residue found by the algorithm must lie not more than $x = m^{1/n}/b^{(n-1)/n}$ from the exact fraction $\frac{a}{b}m$. Since $b \approx m^{1/(n^2-1)}$, $x = m^{1/(n+1)}$ and $\Delta = m^{(n^2-3)/(n^2-1)}$. It can be checked that for $n = 2$, $x = \Delta = m^{1/3}$, and the algorithm always finds the solution, but when $n > 2$, the algorithm fails if the clear text is farther from the fraction than x .

References

1. Batut, C., Belabas, K., Bernardi, D., Cohen, H. and Olivier, M. PARI-GP Number Theory Package. <ftp://megrez.math.u-bordeaux.fr/pub/pari/>.
2. Brickell, E. F. and Odlyzko, M. Cryptanalysis: A survey of recent results. Proc. of the IEEE **76** (1988) 578–593
3. Cohen, H.: A Course in Computational Algebraic Number Theory, Springer-Verlag, Berlin (1993)
4. Coppersmith, D. Small solutions to polynomial equations, and low exponents RSA vulnerabilities. J. Cryptology **10** (1997) 233–260
5. Hastad, J. Solving simultaneous modular equations of low degree. SIAM J. Comp. **17** (1988) 336–341
6. Lenstra, A. K., Lenstra, H. W. and Lovász, L. Factoring polynomials with integer coefficients. Mathematische Annalen, **261** (1982) pp. 513–534
7. Peralta, R.: A quadratic sieve on the n -dimensional cube. In: Brickell, E. F. (ed.): Advances in Cryptology, CRYPT'92. Lecture Notes in Computer Science, Vol. 740. Springer-Verlag, Berlin Heidelberg New York (1993) 324–332.
8. Pomerance, C.: The quadratic sieve factoring algorithm. In: Beth, T., Cot, N., and Ingemarsson, I. (eds.): Advances in Cryptology, EUROCRYPT'84. Lecture Notes in Computer Science, Vol. 209. Springer-Verlag, Berlin Heidelberg New York (1985) 169–182.
9. Niven, I., Zuckerman, H.S. and Montgomery, H.L. An Introduction to the Theory of Numbers, John Wiley & Sons, Inc., New York, (1991)
10. Rivest, R. L., Shamir, A. and Adleman, L. A method for obtaining digital signatures and public key cryptosystems. Comm. ACM, **21** (1978) 120–126
11. Silverman, R.D.: The multiple polynomial quadratic sieve. Math. Comp. **48** (1987) 329–339
12. Vallée B.: Generation of elements with small modular squares and provably fast integer factoring algorithms. Math. Comp. **56** (1991) 823–849

APPENDIX

Let the polynomial congruence be $f(i) \equiv 0 \pmod{cb^k}$, for which a solution mod cb is known. The solution can be undertaken following the method given in [9], §2.6 p.86 adapted for this particular case. It can be seen that the method does not hold for the prime or composite character of c or b . Let a_1 be the solution mod cb , then the solution mod cb^2 is of the form $a_2 = a_1 + tcb$ for $0 \leq t < b$. Likewise, the solution mod cb^3 is of the form $a_3 = a_2 + tcb^2$, which can be lifted to a modulus cb^{j+1} by calculating the corresponding t parameter in $a_{j+1} = a_j + tcb^j$. Using Taylor's expansion, we can write

$$f(a_j + tcb^j) = f(a_j) + tcb^j f'(a_j) + t^2 c^2 b^{2j} f''(a_j)/2! + \cdots + t^n c^n b^{nj} f^{(n)}(a_j)/n!$$

for $j \geq 1$, and n being the degree of the polynomial. Note that higher derivatives become identically zero. Reducing the equation mod cb^{j+1} , we obtain

$$f(a_j + tcb^j) \equiv f(a_j) + tcb^j f'(a_j) \pmod{cb^{j+1}}$$

because $f^{(k)}(a_j)/k!$ is always an integer, as is shown in the cited text. Since $f(a_j + tcb^j) \equiv 0 \pmod{cb^{j+1}}$,

$$f(a_j) + tcb^j f'(a_j) \equiv 0 \pmod{cb^{j+1}}$$

and taking into account that a_j is a root of the polynomial mod cb^j , hence $f(a_j)/cb^j$ is an integer, we cancel the term cb^j leading to

$$tf'(a_1) \equiv -\frac{f(a_j)}{cb^j} \pmod{cb},$$

which gives the desired and unique parameter t whenever $f'(a_1)$ is invertible. Since this is the particular case in which we are interested, we can write finally

$$a_{j+1} = a_j - f(a_j) \overline{f'(a_1)} \tag{19}$$

$\overline{f'(a_1)}$ being the multiplicative inverse of $f'(a_1)$ mod cb .

We deal now with the solution of equation (7). Since the modulus $b' = b^j/db^{n-j-1}$, j being the least integer such that $d \mid b^j$, we can take $c = 1$ when $d = 1$ or $c = b^j/d$ otherwise, and solve the equation by iteration of (19) if a solution is known mod cb . The polynomial can be expressed as $f(i) = A_0 + A_1 i + A_2 b i^2 + A_3 b^2 i^3 \cdots + A_{n-1} b^{n-2} i^{n-1}$ with $A_0 = -K$, $A_1 = \frac{n}{d} \beta_{n-1}$, $A_2 = \frac{n(n-1)}{2d} \beta_{n-2}$, $A_3 = \frac{n(n-1)(n-2)}{2 \cdot 3d} \beta_{n-3} \cdots$. The solution is straightforward when $d = 1$, because the polynomial reduced mod b becomes

$$-K + n\beta_{n-1}i \equiv 0 \pmod{b},$$

which gives $a_1 \equiv K \overline{n\beta_{n-1}}$ and $\overline{f'(a_1)} \equiv \overline{n\beta_{n-1}} \pmod{b}$. The iteration of (19) finds the solution mod b^{n-1} . Note that whenever β_{n-1} is coprime with b , a solution exists for any value of K .

There are some cases for which K can not take odd values. Precisely, let $n > 2$, $n = 2^j t$ and $b = 2u$, $j \geq 1$ and t and u odd, then $d = 2^j v$ for an odd v . The polynomial $f(i)$ becomes

$$f(i) = -K + \frac{2^j t}{2^{jv}} \beta_{n-1} i + \frac{2^j t(2^j t - 1)}{2^{j+1} v} 2u \beta_{n-2} i^2 + \cdots.$$

Provided that the values β_k are coprime with b , the polynomial reduced mod 2 becomes $f(i) \equiv -K + i + i^2 \equiv 0 \pmod{2}$, which only has solution for K even. Then, the solution of the polynomial mod b' is also only allowed for K even.

Securing Elliptic Curve Point Multiplication against Side-Channel Attacks

Bodo Möller

Technische Universität Darmstadt, Fachbereich Informatik
`moeller@cdc.informatik.tu-darmstadt.de`

Abstract. For making elliptic curve point multiplication secure against side-channel attacks, various methods have been proposed using special point representations for specifically chosen elliptic curves. We show that the same goal can be achieved based on conventional elliptic curve arithmetic implementations. Our point multiplication method is much more general than the proposals requiring non-standard point representations; in particular, it can be used with the curves recommended by NIST and SECG. It also provides efficiency advantages over most earlier proposals.

1 Introduction

Side-channel attacks on implementations of cryptosystems use observations such as timings [9] or power consumption measurements [10] in order to obtain information that is supposed to be kept secret. In elliptic curve cryptosystems, a particular target for side-channel attacks are algorithms used for point multiplication. The computational task is to compute a product $[e]P$ where P is a point on an elliptic curve $E(\mathbb{F}_q)$ over a finite field and e is a secret positive integer. Point P is usually not secret, and indeed may often be chosen by the attacker. Multiplier e may either be ephemeral or serve as a long-time key.

Elliptic curves used for cryptography are usually required to have a large prime-order subgroup. We denote its order by p and assume that only one order- p subgroup exists. In this setting, cryptographic protocols typically employ only points of order p .

Various countermeasures against power analysis have been proposed that randomise the computation to make the attacker's task harder:

- If projective coordinates are used for representing points (which is usually the case for efficiency reasons, see [1] and [7]), then the representation of the input point can easily be transformed into a random equivalent representation [6].
- $[e]P$ can be expressed as $[e](P - Q) + [e]Q$ where Q is a random point.
- $[e]P$ can be expressed as $[e + np]P$ or $[e - n]P + [n]P$ where n is a random integer.

However, while these countermeasures may provide protection against differential side-channel analysis (i.e. attacks requiring correlated measurements from

multiple computations), they are not likely to provide sufficient protection if simple side-channel analysis is possible (i.e. the direct interpretation of measurements from a single computation). Note that straightforward implementations of elliptic curve systems are particularly vulnerable to simple side-channel analysis because the usual point addition algorithm cannot be used for point doubling: As adding and doubling require different algorithms, the bits of e may be plainly visible in a power consumption trace if the double-and-add algorithm is used for point multiplication. Improved point multiplication algorithms such as the m -ary or sliding window method or their counterparts using signed digits or signed windows [12] will obscure e to some degree, but may still reveal plenty of information.

For these reasons, Liardet and Smart [11] and Joye and Quisquater [8] have proposed reducing information leakage by using special representations of points of certain elliptic curves over prime fields such that a single formula can be used both for addition and doubling. This approach induces some performance penalty, however (with the usual algorithms, point doubling is much faster than addition). Okeya and Sakurai [16] take a different approach by using a variant of Montgomery's point multiplication algorithm [13] for suitable curves over odd-characteristic fields, where the usual group operations are replaced by certain special operations working with triplets of points with y -coordinates omitted, allowing to implement a quite efficient variant of the binary point multiplication method that does not readily reveal the bits of the multiplier.

We will not discuss the details of these methods. A common disadvantage is that each of them requires specifically selected elliptic curves: All curves suitable for [11] (curves with Jacobi form) have group order divisible by 4; curves suitable for [8] (curves with Hesse form) have group order divisible by 3; and curves suitable for [16] (curves with Montgomery form) again have group order divisible by 4. None of these methods is applicable to the NIST and SECG recommended curves given in [14] and [4], whose use is often encouraged in order to ease interoperability.

We propose an alternative approach without a limitation to specifically chosen curves (and indeed without a limitation to curves over fields of odd characteristic): We accept that doublings are more efficient than additions, but we use these two in a uniform pattern. This makes our method largely independent of the particular point representation employed. We will see that in terms of field multiplications our method with the usual Jacobian projective coordinates is typically more efficient than the Liardet-Smart and Joye-Quisquater proposals.

Our approach requires more additions than the standard 2^w -ary point multiplication algorithm, of which it is a variant, but it does not involve dummy additions in the sense of throwing away results of point operations. Dummy additions provide an immediate way to achieve a fixed pattern of point doublings and point additions [6, section 3.1]; however, as we will explain in section 2, it appears prudent to avoid them.

The method may fail in some cases in that an addition step may turn out to be a point doubling or may involve the point at infinity (which both requires special

treatment and is potentially clearly visible through side channels). However, we will show that the probability of this happening is negligible if multipliers are appropriately selected: Randomly chosen e is safe.

Multiple side-channel attack countermeasures can be combined; the point multiplication method proposed in the current paper can be used together with one or multiple of the randomisation methods mentioned above. In particular, randomised projective coordinates should be used (cf. [16]).

In section 2, we discuss assumptions of the security model underlying our side-channel attack countermeasure and look into certain implementation details. Section 3 presents our approach for implementing elliptic curve point multiplication with security against side-channel attacks. Section 4 provides an efficiency comparison between our method and the methods of [11], [8], and [16].

2 Security against Side-Channel Attacks

The goal to achieve security against side-channel attacks warrants some notes on the model to be used for the security analysis. While a complete model of all side-channel information is hardly ever attainable (and, in any case, would be closely tied to a specific implementation), it is possible to describe the workings of the point multiplication algorithm in enough detail to obtain reasonable assurance that information leakage will not be useful to an attacker. Before presenting the actual point multiplication method in section 3, we discuss desired features and how to achieve them. The most prominent item to consider are elliptic curve point operations. We enumerate special cases of point operations that should be avoided (section 2.1). At a lower level, we can focus on the individual field operations used to implement point operations. In section 2.2, we show that point multiplication implementations intended to be secure against side-channel attacks should use randomised projective coordinates and should use certain extended point representations; we also explain why inserting dummy point additions to achieve uniform behaviour would be questionable.

2.1 Elliptic Curve Point Operations

As different algorithms are usually required for point doubling and point addition, we assume that side-channel data reveals the sequence in which these operations take place. Thus the point multiplication algorithm should use doublings and additions in a uniform pattern independent of the specific multiplier. We also have to take into account certain special cases that must be expected to be visible through side channels when they occur:

- Point doubling $[2]A$ requires conditional statements for the case that A is the point at infinity or that A is a point of order two. If these cases are avoided, then, expressed in field operations, point doubling runs as a fixed routine.
- Point addition $A + B$ requires conditional statements for the case that one of the points is the point at infinity, or that A coincides with B , or that one

point is the inverse of the other. For other cases, it too can be implemented as a fixed routine.

Details of the sequences of field operations used for point doubling and point addition depend on the underlying field (odd characteristic vs. characteristic 2) and the choice of point representations (e.g. either affine coordinates or one of multiple styles of projective coordinates, cf. [5]), so implementations may vary widely. The essential observation is that the respective algorithm always behaves the same as long as the above special cases are avoided.

2.2 Field Operations

While it is reasonable to assume that side-channel observations for, say, a field multiplication do not immediately reveal the factors involved, it would not be prudent to assume that all multiplications look the same to an attacker. This is why randomised projective coordinates are useful ([6], [16]): If, e.g., Jacobian projective coordinates are used, triplets (X, Y, Z) with $Z \neq 0$ represent affine points $(X/Z^2, Y/Z^3)$; then for any field element $\lambda \neq 0$, $(\lambda^2 X, \lambda^3 Y, \lambda Z)$ is a representation of the same point on the curve. If λ is secretly chosen at random, none of the coordinates of the new representation will be predictable to the attacker. Point doubling or point addition using projective coordinates results in a point represented with a Z -coordinate that is the product of the Z -coordinate(s) of the input point(s) and a short polynomial involving one or more other coordinates of the input points; thus the output point is again in a randomised representation.

When use of randomised projective coordinates prevents the attacker from guessing the specific inputs to field operations (and thus from directly verifying, for example, which of several known points are currently being added), the attacker may still be able to recognise if the same field operation with the same input values is performed multiple times. Even if these values are not known, this may leak essential information, so we want to avoid it. We now show what should be taken care of.

Many algorithms for point multiplication $[e]P$, including the one that we will examine in section 3, can be roughly outlined as follows:

[Precomputation stage.] First, independently of the specific multiplier e , certain small multiples of P are computed and stored in a table.

[Evaluation stage.] Second, the product $[e]P$ is evaluated as follows: A variable A is initialized to one of the table values; then, many times, A either is doubled or a table value is added to A , replacing the previous value of A . Finally, A contains the result $[e]P$.

For algorithms using this structure, each entry of the precomputed table should contain not only the representation of the point, but additionally any field elements that would have to be computed each time the point is added to A . If Jacobian projective coordinates are used, this means that (X, Y, Z, Z^2) should be stored rather than just (X, Y, Z) (cf. the addition formulas in [1, Chapter IV.2] or [7]). If such *extended point representations* are not used for the precomputed

table, the attacker may be able to tell which point additions use the same table value.

Most point multiplication algorithms of the above form can achieve a fixed pattern of doublings and additions only if dummy additions are inserted into the evaluation stage. That is, sometimes a table value is added to A , but the result is thrown away and A is left unchanged. The problem with dummy additions is the the attacker may be able to tell that their result is ignored: For Jacobian projective coordinates, each point operation requires squaring the Z -coordinate of A ; if A is not changed, then two consecutive point operations involve the very same squaring. The point multiplication algorithm presented in section 3 achieves uniform behaviour without resorting to dummy additions.

Finally, we show how to use randomised projective coordinates in practice. It is possible, but inefficient, to randomise the point representation after each point operation. (If this is done, dummy additions are no longer a problem.) For point multiplication algorithms of the above form, it is more practical to use randomisation just twice or once: If the precomputed table of multiples of P is stored in projective coordinates, then the representation of P should be randomised before the table is computed. Also at the beginning of the second stage, after the initial value has been assigned to A , the representation of A should be randomised. If the table of multiples of P is stored in affine coordinates (to speed up the evaluation stage by using mixed addition of affine and projective points [5]), then the first randomisation obviously is not necessary.

3 Multiplier Recoding Providing Resistance against Side-Channel Attacks

We show how to perform point multiplication $[e]P$ in a way such that doublings and additions occur in a fixed pattern in order to provide resistance against side-channel attacks. Section 3.1 describes an algorithm for recoding the multiplier e into a special signed-digit encoding. In section 3.2, we discuss the multiplication algorithm implied by this encoding. Section 3.3 shows that, unless e is ill-chosen, this point multiplication algorithm indeed limits information leakage as intended.

3.1 Recoding Algorithm

Let the positive integer e be given in 2^w -ary digits where $w \geq 2$ is a small integer: That is,

$$e = \sum_{i=0}^{k'} b'_i \cdot 2^{wi}$$

with $b'_i \in \{0, 1, \dots, 2^w - 1\}$. We demand that k' be chosen minimal, i.e. $b'_{k'} \neq 0$. For $i > k'$, we define that $b'_i = 0$.

We will show how to convert this into a representation

$$e = \sum_{i=0}^k b_i \cdot 2^{wi}$$

such that $b_i \in \{-2^w, 1, 2, \dots, 2^w - 1\}$. This means that we disallow digit value 0 and instead introduce the new value -2^w . Intuitively, the recoding algorithm replaces 0 digits by -2^w and increments the next more significant digit to adjust the value. It is easy to see that b_k must be positive (otherwise e would be negative) and that the representation of e needs to grow by at most one digit, i.e. $k = k'$ or $k = k' + 1$.

We express the recoding algorithm recursively, using auxiliary values c_i and t_i such that $0 \leq c_i \leq 2$ and $0 \leq t_i \leq 2^w + 1$: Let

$$c_0 = 0,$$

and for $i = 0, \dots, k' + 1$, let

$$t_i = b'_i + c_i$$

and

$$(c_{i+1}, b_i) = \begin{cases} (1, -2^w) & \text{if } t_i = 0 \\ (0, t_i) & \text{if } 0 < t_i < 2^w \\ (2, -2^w) & \text{if } t_i = 2^w \\ (1, 1) & \text{if } t_i = 2^w + 1. \end{cases}$$

Note that we always have $c_{i+1} \cdot 2^w + b_i = t_i$.

If $b_{k'+1} \neq -2^w$, then $e = \sum_{i=0}^{k'+1} b_i \cdot 2^{wi}$; in this case, we set $k = k' + 1$. Otherwise, we have $e = \sum_{i=0}^{k'} b_i \cdot 2^{wi}$ and $b'_k \neq -2^w$, and we set $k = k'$.

Observe that if $b_k = 1$ (and $k > 0$), then $b_{k-1} \neq -2^w$.

As a measure against side-channel attacks on the recoding algorithm itself, assignments can be implemented by table lookups instead of using conditional statements.

Storing the converted representation of e requires almost no additional memory compared with the original representation: Digits with value -2^w can be encoded as a pattern of bits all set to zero. If w is understood, then this new representation can be stored as an ordinary integer. (Leading 0 digits can simply be ignored because b_k is never -2^w .) This integer is at most two bits longer than e ; the maximum possible length growth occurs if $k = k' + 1$ and $b_k = 2$.

It may be desirable to ensure that the encoded form has a predetermined maximum index k . For our point multiplication application, if kw is large enough compared with the binary length of p , this is easy to do: If necessary, adjust e by adding p or a multiple of p .

If a random multiplier is required and a uniform distribution is not essential, then random bits can be used directly to generate the recoded form of e . In this case, too, it should be ensured that $b_{k-1} \neq -2^w$ if $b_k = 1$.

3.2 Point Multiplication Algorithm

Remember that we want to compute $[e]P$ where point P should have order p , p being a large prime that divides the order of the elliptic curve $E(\mathbb{F}_q)$. For our point multiplication algorithm to work as intended, $\text{ord}(P)$ may be any positive multiple of p .

If point P can be chosen by the attacker, we must be aware that it might have incorrect order. In case that $\#E(\mathbb{F}_q) = p$, then (besides testing that P is indeed a point on the curve) we just have to verify that P is not the point at infinity, \mathcal{O} . Otherwise, we need an additional sanity check. Let h be the cofactor of the elliptic curve, i.e. the integer $\#E(\mathbb{F}_q)/p$. Curves used for cryptography are usually selected such that h is small, e.g. $h \leq 4$ as required by [3]. Thus $[h]P$ can be computed quickly; if it is not \mathcal{O} , then we know that p divides $\text{ord}(P)$.

If P has incorrect order, computing $[e]P$ must be rejected. Otherwise, given a representation

$$e = \sum_{i=0}^k b_i \cdot 2^{wi}$$

of e as determined by the recoding algorithm of 3.1, $[e]P$ can be computed as follows:

Assume that points $P, [2]P, [3]P, \dots, [2^w - 1]P, [-2^w]P$ have been stored in a table. Let $e_j = \sum_{i=j}^k b_i \cdot 2^{w(i-j)}$ for $j = 0, \dots, k$. The goal is to compute $[e_0]P = [e]P$. We can determine $[e_k]P = [b_k]P$ simply by table lookup. Then, to compute $[e_j]P$ for j going down to 0, we can use that $e_j = 2^w \cdot e_{j+1} + b_j$ and thus $[e_j]P = [2^w]([e_{j+1}]P) + [b_j]P$, where $[b_j]P$ again is available by table lookup. This is essentially the 2^w -ary algorithm for computing $[e]P$, except that we use an unusual set of digits.

The procedure is given in algorithm 3.2. We show a high-level description of the algorithm; as explained in section 2.2, implementations should use randomised projective coordinates, and values computed in the precomputation stage should be stored in extended point representation. The algorithm requires exactly $2^{w-1} + kw$ point doublings and $2^{w-1} - 1 + k$ point additions. (Inverting $[2^w]P$ to obtain $[-2^w]P$ is essentially free in elliptic curves.)

3.3 Uniformity of the Point Multiplication Algorithm

It is apparent that algorithm 3.2 uses doublings and additions in a regular pattern. As discussed in section 2, to show that the algorithm has uniform behaviour and thus is suitable for limiting information leakage during the computation of $[e]P$, we also have to show that the following special cases of point doubling and point addition can be avoided:

- $[2]\mathcal{O}$
- $[2]A$ where $\text{ord}(A) = 2$
- $A + \mathcal{O}$ or $\mathcal{O} + B$
- $A + A$
- $A + (-A)$

We have required that $\text{ord}(P)$ be a positive multiple of p . Without loss of generality, here we may assume that $\text{ord}(P) = p$. (Otherwise, multiply P by the cofactor $h = \#E(\mathbb{F}_q)/p$ to obtain a point of order p . In the algorithm performed for P , the above special cases can occur only when one of them would occur

Algorithm 1 Compute $[\sum_{i=0}^k b_i \cdot 2^{wi}]P$

```

{Precomputation stage}
 $P_1 \leftarrow P$ 
for  $n = 2$  to  $2^w - 2$  step 2 do
     $P_n \leftarrow [2]P_{n/2}$ 
     $P_{n+1} \leftarrow P_n + P$ 
     $\{P_n = [n]P, P_{n+1} = [n+1]P\}$ 
end for
 $P_{-2^w} \leftarrow -[2]P_{2^w-1}$ 
 $\{P_{-2^w} = [-2^w]P\}$ 

{Evaluation stage}
 $A \leftarrow P_{b_k}$ 
for  $j = k - 1$  down to 0 do
     $A \leftarrow [2^w]A$ 
     $A \leftarrow A + P_{b_j}$ 
end for
return  $A$ 

```

in the algorithm performed for $[h]P$; in particular, $\text{ord}(A) = 2$ would imply $[h]A = \mathcal{O}$, so points of order 2 are not an issue if we can show that the point at infinity can be avoided.) Then, as 2^w can be expected to be much smaller than p , all precomputed points $P_{b_j} = [b_j]P$ in algorithm 3.2 will be of order p . Thus, initially we have $A \neq \mathcal{O}$; and as long as we avoid additions of the form $A + (-A)$, it will stay like this. So what remains to be checked is whether in the evaluation stage of algorithm 3.2 we can avoid that $A = [b_j]P$ or $A = [-b_j]P$ before an addition takes place.

With e_j defined as in section 3.2, the point addition step in the second loop of algorithm 3.2 performs the point addition $[2^w \cdot e_{j+1}]P + [b_j]P$. Thus we are save if

$$2^w \cdot e_{j+1} \not\equiv \pm b_j \pmod{p}.$$

Since $|e_j| \leq 2^{(1+k-j)w}$ and $|b_j| \leq 2^w$, for many indices j we have $|2^w \cdot e_{j+1}| + |b_j| < p$, meaning that reduction modulo p does not matter and it suffices to check whether $2^w \cdot e_{j+1} \neq \pm b_j$.

The largest index to consider is $j = k-1$: Can we be sure that $2^w \cdot e_k \neq \pm b_{k-1}$? Indeed we can, as $e_k = b_k$ and if $b_k = 1$, then $b_{k-1} \neq -2^w$ (see section 3.1). It follows that $e_j \geq 2^{(k-j)w}$, which shows that the incongruence is satisfied for indices $j < k-1$ too as long as we do not have to consider reduction modulo p .

For indices j so small such that this modular reduction is relevant, we argue that if e is sufficiently random, then the probability of picking an e such that the above incongruence is not satisfied can be neglected: It is comparable to the probability that $e \bmod p$ can be guessed in a modest number of attempts, which can be presumed to be practically impossible.

4 Efficiency Comparison

We show that using the point multiplication method of section 3 with the usual Jacobian projective representation of points is typically more time-efficient than the method of Joye and Quisquater [8] and the method of Liardet and Smart [11]. The method of Okeya and Sakurai [16], however, has performance advantages (the downside being that it requires specific curves).

With the Joye-Quisquater proposal, 12 multiplications in the underlying field are needed for each addition or doubling [8]. (The Liardet-Smart proposal requires 16 field multiplications for each addition or doubling [11] and thus is computationally more expensive, so we omit in the remainder of this comparison.) This means that for computing $[e]P$, the expected number of field multiplications per bit of e is more than 12. For a more precise estimate, assume that the point multiplication algorithm based on signed windows [12] is used (one of the most efficient general point multiplication algorithms) and that $2^w - 1$ points are precomputed. Then the number of field multiplications per bit is about $(1 + \frac{1}{w+3}) \cdot 12$, not including the precomputation effort. For multipliers whose length ℓ is between 120 and 336 bits, the expected number of operations required by the signed window method is minimised when $2^3 - 1 = 7$ points are precomputed; the expected number of field operations then is about

$$\left(7 + \frac{7}{6} \cdot \ell\right) \cdot 12 = 84 + 14 \cdot \ell$$

including precomputation.

For elliptic curve cryptography over fields of odd characteristic, curves are usually chosen such that a doubling can be done in 8 field multiplications and an addition can be done in 16 field multiplications using Jacobian projective coordinates [7]. (Further speed-ups are possible by using mixed coordinates for faster point addition; see [5] and [2].) With our method of section 3, using an extended point representation for precomputed points as explained in section 2.2, one additional field multiplication is required for each precomputed point, but only 15 field multiplications are needed for each evaluation stage point addition. Thus, if we use $w = 4$, we need $8 + \frac{1}{4} \cdot 15 = 11.75$ field multiplications for each bit of e (not including the effort to precompute 15 points). This is less than 12, so for sufficiently long e this method will be faster than the Joye-Quisquater method. Choosing $w = 4$ minimises the number of field multiplications for scalars of length ℓ between 84 and 277 bits; it is about

$$192 + 11.75 \cdot \ell,$$

including $2^3 \cdot 8 + (2^3 - 1) \cdot 16 + 2^4 = 192$ multiplications for precomputation using extended point representation (but neglecting the small additional cost for randomising projective coordinates). Compared with the Joye-Quisquater approach, this is nearly 10% faster even for multipliers as short as 120 bits.

The method of [16] for curves having a Montgomery form is more efficient than any of the other methods: It needs only 11 field multiplications per bit [15], and as it is directly based on the binary point multiplication algorithm, it does not involve any precomputation.

5 Conclusion

We have presented a method for elliptic curve point multiplication that can be shown to provide security against side-channel attacks. The algorithm uses a special signed-digit encoding to ensure that point doublings and point additions occur in a uniform pattern. No dummy additions are required; implementing the method using randomised projective coordinates and storing precomputed points in extended point representation limits information leakage to a minimum. While the method of Okeya and Sakurai [16] for curves with Montgomery form is more efficient, the approach of the current paper is much more general: Unlike various previous proposals, it is applicable to the recommended curves of [14] and [4].

References

1. BLAKE, I. F., SEROUSSI, G., AND SMART, N. P. *Elliptic Curves in Cryptography*, vol. 265 of *London Mathematical Society Lecture Note Series*. Cambridge University Press, 1999.
2. BROWN, M., HANKERSON, D., LÓPEZ, J., AND MENEZES, A. Software implementation of the NIST elliptic curves over prime fields. In *Progress in Cryptology – CT-RSA 2001* (2001), D. Naccache, Ed., vol. 2020 of *Lecture Notes in Computer Science*, pp. 250–265.
3. CERTICOM RESEARCH. Standards for efficient cryptography – SEC 1: Elliptic curve cryptography. Version 1.0, 2000. Available from <http://www.secg.org/>.
4. CERTICOM RESEARCH. Standards for efficient cryptography – SEC 2: Recommended elliptic curve cryptography domain parameters. Version 1.0, 2000. Available from <http://www.secg.org/>.
5. COHEN, H., ONO, T., AND MIYAJI, A. Efficient elliptic curve exponentiation using mixed coordinates. In *Advances in Cryptology – ASIACRYPT ’98* (1998), K. Ohta and D. Pei, Eds., vol. 1514 of *Lecture Notes in Computer Science*, pp. 51–65.
6. CORON, J.-S. Resistance against differential power analysis for elliptic curve cryptosystems. In *Cryptographic Hardware and Embedded Systems – CHES ’99* (1999), C. K. Koç and C. Paar, Eds., vol. 1717 of *Lecture Notes in Computer Science*, pp. 292–302.
7. INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS (IEEE). IEEE standard specifications for public-key cryptography. IEEE Std 1363-2000, 2000.
8. JOYE, M., AND QUISQUATER, J.-J. Hessian elliptic curves and side-channel attacks. In *Cryptographic Hardware and Embedded Systems – CHES 2001 [Pre-]Proceedings* (2001), C. K. Koç, D. Naccache, and C. Paar, Eds., pp. 412–420.
9. KOCHER, P. C. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In *Advances in Cryptology – CRYPTO ’96* (1996), N. Koblitz, Ed., vol. 1109 of *Lecture Notes in Computer Science*, pp. 104–113.
10. KOCHER, P. C., JAFFE, J., AND JUN, B. Differential power analysis. In *Advances in Cryptology – CRYPTO ’99* (1999), M. Wiener, Ed., vol. 1666 of *Lecture Notes in Computer Science*, pp. 388–397.
11. LIARDET, P.-Y., AND SMART, N. P. Preventing SPA/DPA in ECC systems using the Jacobi form. In *Cryptographic Hardware and Embedded Systems – CHES 2001 [Pre-]Proceedings* (2001), C. K. Koç, D. Naccache, and C. Paar, Eds., pp. 401–411.

12. MIYAJI, A., ONO, T., AND COHEN, H. Efficient elliptic curve exponentiation. In *International Conference on Information and Communications Security – ICICS '97* (1997), Y. Han, T. Okamoto, and S. Qing, Eds., vol. 1334 of *Lecture Notes in Computer Science*, pp. 282–290.
13. MONTGOMERY, P. L. Speeding the Pollard and elliptic curve methods of factorization. *Mathematics of Computation* 48 (1987), 243–264.
14. NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY (NIST). Digital Signature Standard (DSS). FIPS PUB 186-2, 2000.
15. OKEYA, K., KURUMATANI, H., AND SAKURAI, K. Elliptic curves with the Montgomery-form and their cryptographic applications. In *Public Key Cryptography – PKC 2000* (2000), H. Imai and Y. Zheng, Eds., vol. 1751 of *Lecture Notes in Computer Science*, pp. 238–257.
16. OKEYA, K., AND SAKURAI, K. Power analysis breaks elliptic curve cryptosystems even secure against the timing attack. In *Progress in Cryptology – INDOCRYPT 2000* (2000), B. K. Roy and E. Okamoto, Eds., vol. 1977 of *Lecture Notes in Computer Science*, pp. 178–190.

A Flexible Role-Based Access Control Model for Multimedia Medical Image Database Systems

Sofia Tzelepi and George Pangalos

Informatics Laboratory, Computers Division, General Department, Faculty of Technology,
Aristotelian University, Thessaloniki 54006, Greece
{tzelepi, gip}@eng.auth.gr

Abstract. Most of the work on multimedia medical images security until now has focused on cryptographic approaches. While valuable, cryptography is not enough to control access to images. Therefore additional protection approaches should be applied at a higher level. Role-based access control (RBAC) is a good candidate to provide access control in a multimedia medical image DBMS. However, in a multimedia medical image DBMS, specifications of image access rights are often based on the semantic content of the images, the attributes of the user accessing the image, the relationship between the user and the patient whose images are to be accessed and the time. Unfortunately, RBAC cannot be used to handle the above requirements. In this paper we describe an extended RBAC model by using constraints in the specification of the Role-Permission relationship. The proposed access control model preserves the advantages of scaleable security administration that RBAC-style models offer and yet offers the flexibility to specify very fine-grained, flexible, content, context and time-based access control policies.

1 Introduction

In many health care information systems medical images are an important part of the multimedia medical patient record. Most of the work on multimedia medical images security until now has focused on cryptographic approaches [1], [2], [3]. While valuable, cryptography is not enough to control access to medical images. Cryptography can only control secrecy and authentication aspects, but cannot handle for example different types of access by different users, fine-grained restrictions at the level of individual users and specific images, content, context and time-based access to images [4]. Therefore additional approaches should be applied at a higher level. The health care information systems are generally characterized by users with a diverse set of qualifications and responsibilities that can naturally be mapped to various roles. As such, it appears that role-based access control (RBAC) is a good candidate to provide access control, since roles accurately describe which types of people need access to certain types of objects.

Role-based access control (RBAC) is proposed and studied as an alternative for mandatory (MAC) and discretionary (DAC) access control approaches. In RBAC, it is possible to simplify the complicated form of an organization's access control policy. Access decisions are based on the roles, which is part of an organization. RBAC is a non-discretionary access control in which the system administrator allows the role's permissions to the user by defining user, role, and permission. The system administrator divides roles according to operations in an organization. The administrator of the system or organization gives access permissions to roles and users are endowed with roles according to their responsibility and obligation. Users who are granted a role in system can manage their works with their role permissions. In case of changing access control policy, the system supervisor easily can grant a new permission or can eliminate the existing permission to the role. Because access permissions are granted to roles (permissions are associated with roles), not to users, it is possible to manage access control policy more efficiently. There are many variations of RBAC, but the basic architecture of RBAC is that permissions are assigned to roles (not directly to users) and roles are assigned to users [5].

The notion of roles is an important factor in authorization rules, but in a multimedia medical image database system context in order to be effective it has to be used in conjunction with the following information:

- Semantic content of the images: image access is naturally described in terms of its semantic contents, for example, all images presenting a cancer of the lung must not be made available to physicians who are accessing information from no trust domain.
- Domain: what domain of the health system a particular caregiver works for. For example, medical images belong to certain departments and are not accessible by certain physicians, or a physician may be permitted to access only medical images of his/her subordinates and their subordinates, recursively.
- Location: where the user is accessing information services from. Location information is used in several types of authorization rules. One type uses location to identify the trust domain where the user is accessing information services from. A reasonable policy would deny access to any sensitive information to anyone accessing it from such areas. Location can also be used to derive the emergency level of access. A policy can allow read access to all images of all patients for any user assigned to the role physician and accessing the information from an emergency room.
- Time: time constraints specify the validity periods for a policy.
- Relationship: what is the relationship between the user and the patient whose images are to be accessed. Some types of relationships that need to be managed in the healthcare context are: patient's primary care provider; admitting, attending, referring, or consulting physician of a particular patient; part of the patient care team; healthcare staff explicitly assigned to take care of the patient; patient's immediate family; patient's legal counsel or guard; personal pastoral care provider.

Unfortunately, RBAC cannot be used to handle the above requirements [6]. In order to overcome this problem, in this paper we propose an extended role-based access con-

tol model by considering, in the specification of the Role-Permission relationship phase the constraints which must be satisfied in order for the holders of the permission to use those permissions. The use of constraints allows role-based access control to be tailored to specify very fine-grained, flexible, content, context and time-based access control policies. The proposed access control model preserves the advantages of scalable security administration that RBAC-style models offer and yet offers the flexibility to specify complex access restrictions based on the semantic content of the images, the attributes of the user accessing the image, the relationship between the user and the patient whose images are to be accessed and the time.

A subset of Object Constraint Language (OCL) [7] is used for specifying constraints. In the development of content-based constraints a simplified medical image model for describing the semantic content of a medical image is used. The medical image can be viewed as pairs of iso-semantic regions and signals in respect with an anatomic and a pathological model [8]. Moreover, medical images are associated with complementary textual patient information.

The rest of this paper is structured as follows. Section 2 introduces related work and contrasts it with our work. The proposed RBAC and the detailed specification of its components are described in section 3. Section 4 introduces the medical image data model. Section 5 presents the access control mechanisms and the algorithm proposed in this paper. Section 6 introduces the access control architecture and section 7 concludes the paper.

2 Related Work

As mentioned above, one of the problems of applying RBAC to multimedia medical image DBMS is the specification and enforcement of fine-grained access control at the level of individual users and specific images [3]. For example, just because the doctor's role enables a set of accesses to medical images does not mean that the doctor's role should provide access to all medical images. A doctor can only access the medical images for those patient currently assigned to this doctor. There have been several approaches for creating an instance level policy for roles by using the notion of team-based access control (TMAC) [3] or by introducing parameterized roles to RBAC models [4], [5].

An alternative approach to specify and enforce fine-grained access control at the level of individual users and specific images is proposed in this paper by considering, in the specification of the Role-Permission relationship phase, the constraints which must be satisfied in order for the holders of the permission to use those permissions. Furthermore, constraints offer also the ability to specify complex access restrictions based on the semantic content of the images, the attributes of the user accessing the image, the relationship between the user and the patient whose images are to be accessed and the time.

Constraints have also been addressed in [10]. In [10] content-based access control is enforced by simply specifying some constraints against attribute values of data ob-

jects. In contrast, due to the nature of medical images, content-dependent access control for a medical image database system must be based on the semantics of the medical images, rather than on the attributes characterizing them. Medical image attributes often only deal with physical characteristics of the medical images (for example, acquisition device, direction, format,...) and therefore are not significant for access control.

In the development of content-based constraints specification a simplified medical image model for describing the semantic content of a medical image is described in the next section.

3 The Underlying Medical Image Data Model

For content-based access control, medical image databases must have capabilities to recognize and quantitate image content and merge the quantitated image data with textual patient data into a common data model [11]. For the past three decades, the medical image processing community has actively pursued efficient algorithms that can extract and quantify semantic objects from images. Established algorithms can be readily integrated into a multimedia medical DBMS for image segmentation, texture analysis, content extraction and image registration. These can be performed automatically or interactively depending on the difficulty of segmenting and extracting semantic structures.

In this section, we use in the development of our content-based access control model a simplified medical image model introduced in [8]. Our emphasis in this paper will be on the development of our content-based access control model rather than on modeling effectively multimedia medical data. In [8], the medical image model represents

- what is the image (i.e. the pictorial attributes of image)
- what is around the image (i.e. the context)
- and finally what is in the image (i.e. the content)

Figure 1 illustrates the medical image data model. The attributes of the IMAGE entity describe the type of the image (such as the acquisition device (scanner, MRI,...)), the direction (sagittal, coronal, axial) and the format. The attributes of the PATIENT entity describe what is "around" the image such as the data about the patient. Finally, we need a model to describe the semantic content of a medical image.

A medical image usually shows an organic structure, where pathological signals can be detected in special places. The content-based constraints we are concerned with deal with the presence or absence (and eventually with the characteristics) of a referenced object in a particular location of the image. Therefore, the semantic content of a medical image can be described in terms of iso-semantic regions and signs: the regions allow to define and name the location place, the signs are the referenced objects, subjects of the content-based constraints. Our methodology uses an anatomical model

to capture some iso- semantic regions in a medical image, and a pathological model to capture the signs, i.e. the “signals” in their medical definition (see figure 2).

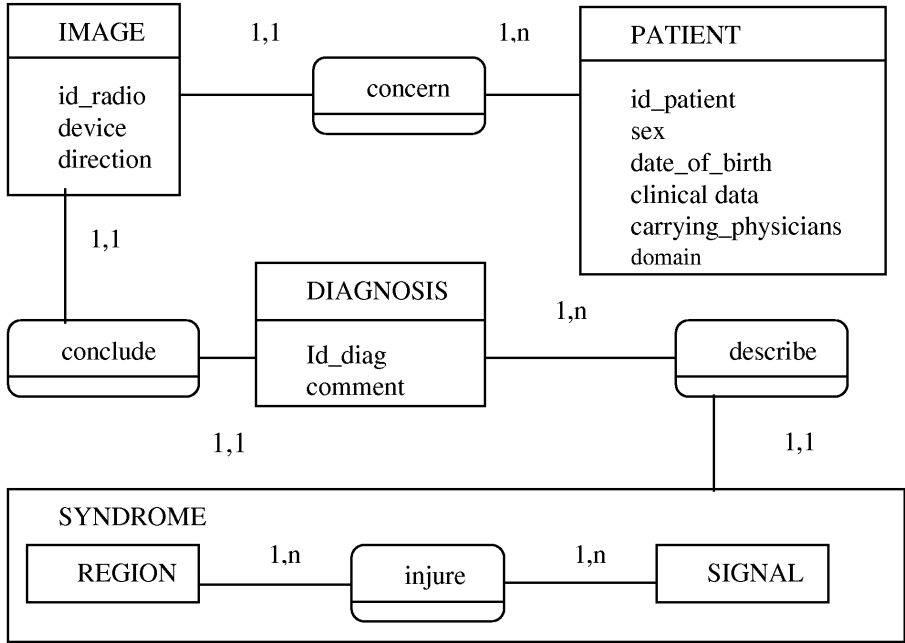


Fig. 1. Medical image model

The anatomical model. In order to describe an organic structure, we propose a decomposition of the organ into substructures that are the iso-semantic regions defined above. These iso-semantic regions are defined by the doctors and are a representation which is near to an anatomic plan of the human body. The edges of the regions define zones that have for example a same nature and/or role. A region is characterized by its name, its nature (vascular, porous), and its role. The location of a signal will not be defined with mathematical coordinates but by giving the name of the region where it appears.

The pathological model. It is necessary to complement the anatomical model with a pathological model. This builds regions where signals can be identified in a non-ambiguous manner. Indeed, each region is associated to the set of signals that is consistent with it. This relationship becomes an integrity constraint when formulating a content-based constraint or adding an image in the database. The signals have attributes for their name, nature, and gravity. To enhance the model, the **SIGNAL** entity can be specialized according to its nature. Overriding then can be needed for example to represent additional attributes as height or weight for a signal types “tumor”.

We use the anatomical model and the pathological model to build a semantic descriptor of an image. Indeed, an image visualizes a list of syndromes, i.e. a list of regions

injured by several signals. Consequently, the semantic descriptor of an image is an entity SYNDROME, set of pairs (region, set of signals). The syndrome is the transcription of what the doctor can see in the image.

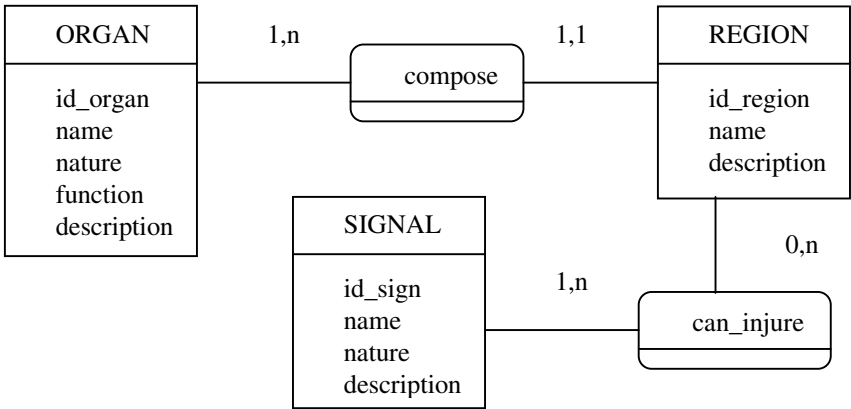


Fig. 2. Semantic content of an image

4 An Extended Role-Based Access Control Model for Multimedia Medical Image Databases

The basic components of a simplified RBAC model are Users, Roles, Permissions, User-Role (U-R) relationship and Role-Permission (R-P) relationship. User is a person who uses the system or an application program in the system. Membership to roles is granted to users based on their obligation and responsibility in the organization. The operation of a user can be carried out based on the user's role.

Role is a set of functional responsibilities within the organization. The system administrator defines roles and assigns them to users. A User-Role (U-R) relationship represents collection of a user and a role.

A permission is the way for the role to access to more than one objects in the system. The terms authorization, access right and privilege are also used in the literature to denote a permission. Permissions are always positive and confer the ability to the holder of the permission to perform some actions in the system. A Role-Permission (R-P) relationship describes which role is assigned to perform what kind of permission in the organization.

In this paper, an extended simplified role-based access control model for multimedia medical image database systems is presented. Two major extensions to the model are introduced. The first extension introduces the notion of user attributes. As mentioned above, there is a need to use user attributes for providing access control. User attributes include among other things the user name, his domain (e.g. position) in the management hierarchy and his location.

A domain is a collection of subjects/objects which have been explicitly grouped together for the purposes of management. It is used to partition the enterprise management scope according to geographical boundaries, administrative departments, etc. For example, subjects/objects inside a department may be grouped in a domain. The concept of a domain is very similar to that of directory in a typical hierarchical file system. The authorization which implies to a domain will, by default, propagate to sub-domains and to the objects within them. User attributes are presented as follows:

$\langle \text{user_id}, \text{user_name}, \text{domain}, \text{location} \rangle$

where, *user_id* is the user identifier, *user_name* is the user name, *domain* is the department of the health system the user works on and *location* is the place where the user is accessing information services from. The attribute *location* is dynamic.

The second extension concerns the Role-Permission relationship. In the proposed model, we consider in the specification of the Role-Permission relationship phase, the constraints which must be satisfied in order for the holders of the permission to use those permissions. In this case, each Role-Permission relationship is a decision rule, which specifies, besides the access modes the holder *s* of the permission is authorized for on image(s) *i*, also the constraints to be satisfied in order for *s* to exercise the access modes. Constraints are based on the semantic content of the images, the attributes of the user accessing the image, the relationship between the user and the patient whose images are to be accessed and the time.

In a multimedia medical database context, the general form of a Role-Permission relationship is 5-tuple

$\langle \text{identifier}, s: r, \{ \text{action} \}, t: \text{target}, \text{constraints}(s, t) \rangle$

According to the above definition, a Role-Permission relationship has the following components:

- identifier: it is used to identify uniquely the permission
- *s*: subject to which the permissions apply
- *r*: role which can process this permission. Subject *s* is authorized for role “*r*”
- action: it is the operation, which is to be processed by role
- *t*: object on which actions are to be performed
- target: it is the object type
- $\text{constraints}(s, t)$: limit the applicability of the permission. Constraints must be satisfied by *s* and *t*.

In a multimedia medical database context low level operations, such as physical read and write operations, are not semantically meaningful for access control in multimedia medical image database. Therefore, in our model we introduce a set of abstract operations that are relevant to the way users actually access medical images.

Users of medical images database go through the following stages. The user first submit a request for a given image. The medical image database server processes the request, and returns to the user either the annotations associated with the image, or the requested image in thumbnail format. The user can then request the display of the full-resolution image in the main display window. Another group of operations for which

access control should also be provided include operations for processing, annotating, deleting the images or for introducing ones. In general, authorizations to perform such operations should be given to few, selected users. For example, Saving, on the main archive, is most always reserved for radiology. Deleting is reserved for the security administrator but only with dual access code security i.e. someone else with security must be present and authorize the deletion. Retrieve, annotate, process can be performed by anyone with the proper access. The different modes of operation, image access privileges, that are provided as part of our model are described in Table 1.

Application example: In the following example we consider a health-care organization security policy. In this example we have a health-care organization composed of several hospitals, and each hospital is structured into some divisions. A primary physician is assigned to a division and he/she can only access medical images for those patients currently assigned in that division and to this doctor. In order to achieve such policy, we define the following role-permission relationship:

$$\{dp1, s: \text{Primary_Physician}, \{view\}, t: \text{Image}, \\ \text{domain_user}(s) = \text{domain}(t) \wedge s \in \text{carrying_physicians}(t)\}$$

The function *domain_user* gives us the domain associated with a user. For instance the domain associated to the primary physician “John Smith” is “hospital1/div2”. Based on the above policy, “John Smith” can only access medical images for those patients currently assigned in the domain “hospital1/div2” and to the doctor “John Smith”.

In the following section we present a detailed specification of the constraints.

Table 1. Image privileges provided by the access control model

Privilege	Meaning
View_annotation	To display the results as the associated annotations only.
View_thumbnail	To display the requested image in thumbnail format. It speeds up the query response time.
Display	To display the full-resolution image in the main display window
Edit_annotation	To edit the annotations of the images
Edit_image	To process, delete or add images to medical images data-base

4.1 Constraints

As said above, an important element of each role-permission relationship is the set of constraints which must be satisfied in order for the holders of the permission to use the permissions. Constraint definitions allow constraints to be separately defined and multiply used. A subset of Object Constraint Language (OCL) is used for specifying constraints which limit the applicability of the permission, for example to a particular time interval or according to the state of the system.

The following are some examples of constraint expressions:

- $\text{domain_user}(s) = \text{domain}(x) \wedge s \in \text{carrying_physicians}(x)$

The function *domain_user* gives us the domain associated with a user. The function *domain* gives us the domain associated with an image. The function *carrying_physicians*(x) gives us the set of physicians associated with patient's image x.

- $\text{"lung"} \in \text{regions}(x) \wedge \text{"cancer"} \in \text{signals}(x, \text{lung})$

This expression denotes all images (x) that present a cancer of the lung. The function *regions* gives us the set of regions associated with an image. The function *signals* gives us the set of signals located in a particular region of an image.

- $\text{"left ventricle"} \in \text{regions}(x) \wedge \text{"tumor"} \in \text{signals}(x, \text{"left ventricle"}) \wedge \text{height}(x, \text{"left ventricle", "tumor"}) \geq 1\text{cm}$.

This expression denotes all images (x) that present a tumor with a height of more than 1 cm on the left ventricle. The function *height* gives us information about the attributes of the signal "tumor".

- $\text{time.between("1600", "1800")}$

This expression limits the policy to apply between 4:00pm and 6:00pm.

5 Access Control

The main goal of the access control mechanism is to verify whether user *u*, trying to access image *i*, using a privilege *p*, under a certain role *r*, is authorized to do so, according to access control restrictions enforced by that role. The access control algorithm is specified in Figure 3.

6 System Architecture

The complete system architecture is depicted in Figure 4. The *authorization manager* is responsible for the full management of both the *Role-Permission relationships base*, *User-Role relationships base* and *User attributes base*. Through the authorization manager, the security administrator can add, modify, or delete User-Role relationships, Role-Permission relationships and User attributes. The *access control manager* implements the access control algorithm in section 5. The *image data manager* is responsible for handling of images. Each time a new image is acquired by the medical image database system, it is first processed by the *image postprocessing manager*, which extracts the semantic content from this image. Information on the semantic content are then stored and used to perform content-based access control restrictions. In particular, in our working prototype we used in the development of our content-based access control model a content-based image interpretation computerized method introduced in [12]. In [12], their work consists of the detection of brain lesions on SPECT images.

We used a relational database management system (Oracle8) to implement our image model as described in section 3 and to store all necessary data (e.g. Role-Permission relationships, User-Role relationships, User attributes). The whole application is written in Pro*C/C++. In our implementation a constraint is evaluated only when a role is activated for a user. A more sophisticated solution might be implemented within an active database system. Trigger mechanisms might be used to dynamically change the user's set of permissions as soon as a specified constraint is no more satisfied.

ALGORITHM 1. Access control Algorithm

INPUT: [1] An access request (u, r, i, p), [2] The User-Role relationship set, [3] The Role-Permission relationship set, [4] The user attributes set

OUTPUT: [1] ACCEPT, [2] REJECT otherwise

METHOD:

```

    If (Is_role_members(u, r)  $\wedge$  Is_role_operations(p, r)) then
        If (evaluation_constraints(u, i, p, r, cn)) then Return (ACCEPT)
        Else Return (REJECT)
    Else Return (REJECT)

```

The function Is_role_members(u, r) returns TRUE if user u is authorized for role r, else return FALSE. The function Is_role_operations(p, r) returns TRUE if operation p is associated with role r, else return FALSE. The function evaluation_constraints(u, i, p, r, cn) returns TRUE if image i and user u satisfies the constraints cn that are associated to the role r, else return FALSE

Fig. 3. Access control algorithm

7 Summary

The RBAC model and mechanism have proven to be useful and effective. Nevertheless, there are many common examples where access decisions must include other factors, in particular, as the semantic content of the images, the attributes of the user accessing the image, the relationship between the user and the patient whose images are to be accessed and the time. In this paper, the above factors are expressed using constraints in the specification of the Role-Permission relationship. The use of constraints allows role-based access control to be tailored to specify very fine-grained, flexible, content, context and time-based access control policies. The proposed access control model preserves the advantages of scaleable security administration that RBAC-style models offer and yet offers the flexibility to specify complex access restrictions.

From our development and implementation experience we are convinced that the proposed model provides significant capabilities to model and implement access control restrictions in a flexible manner, so as to meet the needs of multimedia medical image database management systems.

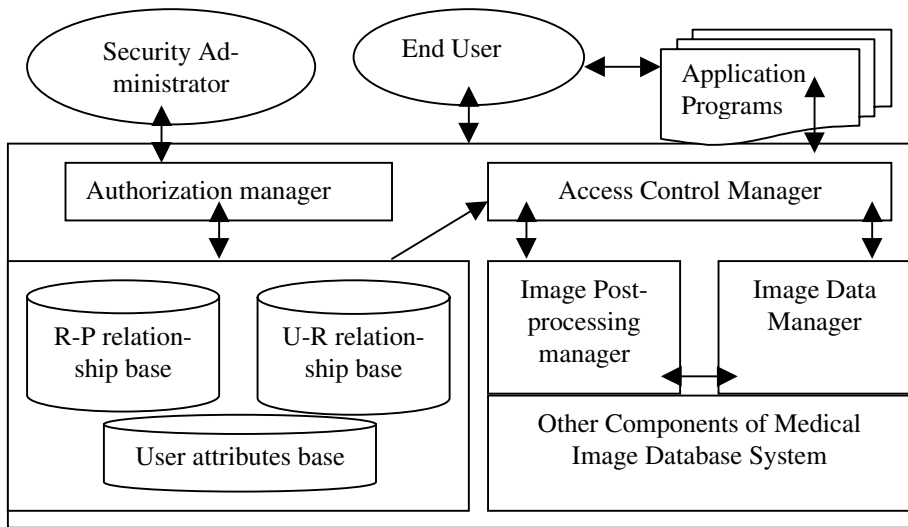


Fig. 4. System architecture for a secure multimedia medical image DBMS

References

1. J. P. Smith, "Authentication of Digital Medical Images with Digital Signature Technology", *Radiology* 1995, 194, pp:771-774.
2. S. T. C. Wong, "A Cryptologic-Based Trust Center for Authenticating Medical Images," *J. American Medical Informatics Assoc.*, Vol. 3, No. 6, Nov./Dec. 1996, pp. 410-421.
3. R. B. Wolfgang and E. J. Delp, "Overview of image security techniques with applications in multimedia systems", *SPIE Conference on Multimedia Networks: Security, Displays, Terminals and Gateways*, Vol. 3228, November 2-5, 1997, Dallas, Texas, pp:297-3308.
4. E. B. Fernandez and K. R. Nair, "An Abstract Authorization System for the Internet", in *Proceedings of the 9th International Workshop on Database and Expert Systems Applications*, 1998.
5. R. Sandhu, E. J. Coynee, H. L. Feinstein, and C. E. Youman, "Role-based access control models", *IEEE Computer*, 29(2), February, 1996.
6. R. K. Thomas, "Team-based access control (TMAC): A primitive for applying role-based access controls in collaborative environments", *ACM RBAC'97*, 1997.
7. Rational Software Corporation, *Object Constraint Language Specification*, Version 1.1, Available at <http://www.rational.com/uml/> September 1997.
8. A. Tchounikine, "Creation and content-based retrieval in a radiological documentary record", in *Proceedings of the 3rd Basque International Workshop on Information Technology*, 1997.
9. L. Giuri and P. Iglio, "Role templates for content-based access control", in *Proceedings of the Second ACM Role-Based Access Control Workshop*, November 1997.
10. E. C. Lupu and M. Sloman, "Reconciling role-based management and role-based access control", in *Proceedings of the Second ACM Role-Based Access Control Workshop*, November 1997.

11. S. T. C. Wong and H. K. Huang, "Design methods and architectural issues of integrated medical image data based systems", *Computerized Medical Imaging and Graphics*, Vol. 20, No 4, pp. 285-299, 1996.
12. E. A. Stamatakis, M. F. Glabus, D. J. Wyper, A. Barnes and J. T. L. Wilson, "Validation of Statistical Parametric Mapping (SPM) in Assessing Cerebral Lesions: A Simulation Study", *NeuroImage* 10, 397-407 (1999).

A Secure Publishing Service for Digital Libraries of XML Documents

Elisa Bertino¹, Barbara Carminati¹, and Elena Ferrari²

¹ Dipartimento di Scienze dell'Informazione, Università di Milano
Via Comelico, 39/41 - 20135 Milano, Italy
{bertino,carminati}@dsi.unimi.it

² Dipartimento di Chimica, Fisica e Matematica, Università dell'Insubria
Via Valleggio, 11 - 22100 Como, Italy
elena.ferrari@uninsubria.it

Abstract. Secure publication over the Internet of XML data is becoming a crucial need as XML is rapidly becoming a standard for document representation and exchange over the Web. Publishing services must have a mechanism that ensures that a user receives all and only those portions of information he/she is entitled to access (for instance those for which the user has paid a subscription fee). Furthermore, such a mechanism must ensure that these contents are not eavesdropped during their transmission from the publishing service to the user. In this paper, we propose an architecture for secure publishing of XML documents. Distinguishing features of our proposal is the flexibility the publishing service offers both in terms of the way users can select the contents they are interested in and in the way the contents are delivered to users. Secure content delivery to users is obtained through the use of different encryption schemes, which ensure that only subscribed users can access the contents managed by the publishing service. In the paper, we first present an overall view of the proposed approach. We then introduce the components of the architecture and the encryption schemes we have developed. Finally, we present algorithms for information delivery to users.

1 Introduction

XML (*eXtensible Markup Language*) [1] is rapidly becoming a standard for document representation and exchange over the Web. An urgent requirement in the Web environment is thus the need for secure publishing services of XML documents. Consider for instance a digital library of XML documents. The digital library must have a mechanism ensuring that a user receives all and only those portions of the library he/she is entitled to access (for instance those for which the user has paid a subscription fee). Furthermore, such a mechanism must ensure that these contents are not eavesdropped during their transmission from the library to the user. Moreover, the service must be flexible in terms of both the way users specify the portions of the library they are interested in receiving and the modes according to which the library contents are released to subscribed

users. To cope with these requirements, in this paper we propose an approach for a secure publishing service of XML documents. Our approach is based on the concept of *package*. A package is defined as a collection of *components*, where each component consists of a portion of information together with a subscription period. The user has complete freedom in defining the information he/she is interested in receiving. A user can subscribe to different packages and each package may contain portions with different subscription periods, according to the user needs. Additionally, users have available a large number of options according to which they can specify the package content. A user can select among a set of pre-defined portions (for instance all the issues of a given newspaper) or can define his/her own portions according to his/her needs. For instance, a user can request to receive, for a specified period of time, all magazines related to a particular topic or the issues of all the newspapers containing articles written by a specified reporter. Additionally, the user can select, for each package, the mode according to which he/she wants to receive its content. Also in this respect, the publishing service supports a variety of package distribution modes and the user can select the most suitable one according to his/her needs. More precisely, the publishing service supports three different distribution modes – *Pull*, *Push*, and *Notify*. If a user selects the pull mode for a given package, he/she has to explicitly request the package portions when needed. By contrast, under the push mode the user does not have to explicitly request the package. Rather the publishing service periodically, or when a modification occurs, sends the modified portions to all the users subscribed to a package to which the portions belong to, without the need of an explicit request. Finally, if the notify mode is selected, the user only receives a notification that a modification in one of his/her package occurs, and he/she has to explicitly request the portion(s) he/she is interested in receiving.

The flexibility provided to the users requires the development of suitable encryption schemes for secure delivery of package contents to subscribers. The schemes must ensure that a user can access the portions of information to which he/she has subscribed for the duration of his/her subscription and that such information is no longer accessible by the user when the subscription expires. To fulfill these requirements we have developed different encryption schemes for the different distribution modes we support. More precisely, in case of pull and notify packages the subscriber receives, upon the completion of the subscription phase, a unique symmetric key [9], which is then used by the publishing service to encrypt the portions of the package returned to the user as answer to an access request. In such a way, only the entitled user is able to decrypt the access request answer because he/she is the only one sharing the encryption key with the publishing service. In case of packages to be delivered under the push mode, we propose a different approach aiming at minimizing the number of keys that need to be generated, by guaranteeing at the same time the security of information delivery. Under the push approach the publishing service broadcasts the same content portion to all the users entitled to access it. Thus, in such a case, the service generates an encryption key for each component. When a user subscribes to a package in *push* mode, he/she receives all the keys associated with the

components belonging to the package. Then, when the publishing service needs to send a portion, it encrypts the portion only once with a session key and sends it to all the users entitled to access it. Such users are those subscribed to a package that contains a component defined over the portion and whose subscription period has not yet expired. Each of those users thus receives the session key properly encrypted with the key associated with the component. As it will be clear through the paper, this approach ensures that the portion can be accessed only by subscribers whose subscription period has not yet expired. Moreover, such an approach ensures that only a limited number of keys need to be generated. Indeed the set of components that can be defined over a source of information is limited, since it depends on the set of possible subscription periods, which is always limited. To the best of our knowledge, the work reported in this paper is the first one aiming at developing a secure publishing service for XML documents. The remainder of this paper is organized as follows. Next section briefly introduces XML. Section 3 presents the overall architecture of the secure publishing service we propose. Section 4 formally introduces the concept of package, whereas Sections 5, 6 and 7 deal with the delivery of information to subscribed users. Section 8 presents an example of package delivery in our framework. Section 9 surveys related work. Finally, Section 10 concludes the paper and outlines future research directions.

2 A Brief Introduction to XML

An XML [1] document is defined in terms of elements and attributes. Elements can be nested at any depth and can contain other elements (*subelements*) in turn originating a hierarchical structure. An element contains a portion of the document delimited by two *tags*: the *start tag*, at the beginning of the element, with the form `<tag-name>`, and the *end tag*, at the end of the element, with the form `</tag-name>`, where *tag-name* indicates the type of the element (*markup*). Attributes can have different types allowing one to specify element identifiers (attributes of type ID), additional information about the element (e.g., attributes of type CDATA containing textual information), or links to other elements in the document (attributes of type IDREF(s)). An example of XML document, modeling the Times newspaper, is given in Figure 1(a). In particular, the Times newspaper consists of a Frontpage (modeled through the `Frontpage` element) containing a leading article (`LeadingArticle` element) and one or more additional short articles (`Paragraph` elements). Moreover, the newspaper contains a politic page (`PoliticPage` element) that includes all the articles related to politics news. Additionally, the newspaper contains an element for the literary page and one for sports news. Another relevant feature of XML is the support for the definition of application-specific document types through the use of *Document Type Definitions* (DTDs). A DTD is composed of two parts: the *element declarations* and the *attribute list declarations*. The element declarations part specifies the structure of the elements contained in the document. In particular, for an element it specifies the order of subelements, whether they are optional

<pre> <Newspaper Title='Times' Date='...'> <Frontpage > <Leading_Article Author='..' Title='..'> </Leading_Article> <Paragraphs > <Paragraph Author='..' Title='..'> </Paragraph Author='..' Title='..'> </Paragraphs> </Frontpage> <Politic_page > <Politic topic='USA' Author='..' Title='..'> </Politic > <Politic topic='EUROPE' Author='..' Title='..'> </Politic > </Politic_page > <Literary_page > <Article topic='Books' Author='..' Title='..'> </Article > <Article topic='Movies' Author='..' Title='..'> </Article > </Literary_page > <Sport_page > <News topic='Soccer' Author='..' Title='..'> </News > <News topic='Basket' Author='..' Title='..'> </News > </Sport_page > </Newspaper > </pre>	<pre> <!DOCTYPE Newspaper[<!ELEMENT Newspaper(Frontpage,Politic_page, Literary_page,Sport_page)> <!ELEMENT Frontpage(Leading_Article,Paragraphs+)> <!ELEMENT Paragraphs (Paragraph+)> <!ELEMENT Politic_page (politic*)> <!ELEMENT Literary_page (Article*)> <!ELEMENT Sport_page (News)> <!ELEMENT Leading_Article (#PCDATA)> <!ELEMENT Paragraph (#PCDATA)> <!ELEMENT Politic (#PCDATA)> <!ELEMENT Article (#PCDATA)> <!ELEMENT News (#PCDATA)> <!ATTLIST Newspaper Title CDATA Date CDATA > <!ATTLIST Leading_Article Author CDATA Title CDATA> <!ATTLIST Paragraph Author CDATA Title CDATA> <!ATTLIST Politic Topic CDATA Author CDATA Title CDATA> <!ATTLIST Article Topic CDATA Author CDATA Title CDATA> <!ATTLIST News Topic CDATA Author CDATA Title CDATA >]> </pre>
(a)	(b)

Fig. 1. (a) An example of XML document and (b) its corresponding DTD

(‘?’), whether they may occur more times (‘*’ or ‘+’ with the usual meaning), and whether subelements are alternative with respect to each other (‘|’). Moreover, it is possible to specify the type of the element data content. This type may be **EMPTY**, if no content is allowed, **ANY** if all kind of content is allowed, or **#PCDATA** if only data content is allowed. The attribute list declarations part specifies, for each element, the list of its attribute names, types, optionality clauses (**#IMPLIED**, to denote an optional attribute, **#REQUIRED**, to denote a mandatory one), and (possibly optional) default values. Figure 1(b) shows the DTD for the XML document in Figure 1(a). In the rest of the paper, we denote with the term *XML Source* the set of XML documents and DTDs, managed by the publishing service.

3 Architecture of the Publishing Service

The architecture of our secure publishing service is depicted in Figure 2. Users are required to register with the publishing service during a mandatory *subscription*

phase. During the subscription phase a user decides which portion(s) of the XML Source he/she is interested in and, for each portion, the user specifies the duration of the subscription. The publishing service presents the user a set of pre-defined portions of the source, corresponding to meaningful information objects (such as for instance, a newspaper, or all the magazines related to a particular topic). Additionally, the user can specify other content portions of the source according to his/her interests. Moreover, the user specifies for each portion the related subscription period. In our approach a portion together with a subscription period is referred to as a *component*.

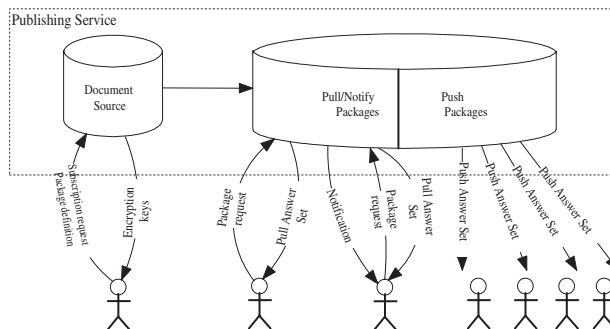


Fig. 2. Publishing service architecture

Additionally, for each component, the user specifies a *distribution mode*, that is, the mode according to which he/she wishes to receive the components. A set of components specified by the same user and with the same distribution mode is collected into a *package*. Thus, a user can subscribe to different packages with different distribution modes, and a package can contain components with different subscription periods. To enhance flexibility, the system supports a variety of package distribution modes, among which the user can select the most appropriate one according to his/her needs. More precisely, the supported package distribution modes are the following:

- *Pull*. If a user subscribes to a package under the pull mode, he/she has to explicitly require the package when needed. Upon a package request by a user, the publishing service first verifies whether the user is entitled to access the requested package. In this case, the system sends the user all the portions of the requested package whose subscription period has not yet been expired and which have been modified after the previous request by the same user.
- *Push*. If the user selects the push mode, he/she does not need to explicitly request the package to the publishing service. Rather, the publishing service upon an update to a portion(s) of the package sends the portion(s) to the user, without the need of an explicit request. The push mode is further specialized into *Push_{on}* and *Push_{off}*. If the *Push_{on}* mode is selected, the

publishing service upon any modification of the source content, sends the modified portions to all the users subscribed to a package to which the portions belong to. By contrast, if the *Push_{off}* mode is selected, the publishing service periodically (for instance once a day) checks which portions of the source have been modified and sends the updated portions to all the users subscribed to a package to which the portions belong to.

- *Notify*. If the user subscribes to a package under the notify mode, the publishing service sends the user a message each time a portion belonging to the package has been modified. Thus, this mode differs from the push mode in that the user is only notified of a modification and can require the modified portions only if he/she is interested in receiving it.

Once a user has selected the distribution mode for a package, the publishing service generates the encryption keys necessary for the package secure delivery to the user. For pull and notify packages, the publishing service generates a unique (symmetric) encryption key which is associated to the whole package. This key is sent to the user upon the subscription phase has been completed and it is used by the publishing service to encrypt the answers to a package request. By contrast, in the push mode, the publishing service sends the content portion(s) to all the users entitled to see them. To limit the number of keys that need to be generated, the publishing service uses a technique based on session keys. In particular, it generates a different key for each component defined over the XML Source. Once a user completes the subscription phase, the publishing service sends the user the keys associated with the components belonging to his/her packages. Then, when the publishing service needs to send a modified portion, it encrypts it with a session key and broadcasts it to all the users subscribed to a package in which one of the components contains such content portion. Then, for each of these components the publishing service encrypts the session key with the key associated with the component.

4 Formal Definitions

In this section we formalize the notions of package, component and portion informally introduced in the previous section. We start by defining the notion of source portion.

Definition 1. (Source Portion). *Let S be an XML Source. A portion over S is defined through a path expression `path_expr`, where `path_expr` is an Xpath [12] expression over an XML document or a DTD belonging to S .*

Note that a path expression over an XML document denotes a portion (or a set of portions) of the document, whereas a path expression over a DTD denotes a portion(s) over all the DTD instances. When a user subscribes to an XML Source, the publishing service presents the user a set of pre-defined portions. The user can then select one or more of those portions or can define additional portions, according to his/her requirements.

Example 1. Consider the XML document in Figure 1(a). Examples of portions that can be created by the publisher service are:

- all the article in literary page,
- all the article in the politic page,
- the Times Frontpage.

The last portion is specified through the Xpath expression
 “//Newspaper[@Title=“Times”]/Frontpage/node()”.

In the following, we denote with *Portions* the set of all the portions defined over the XML Source by the users subscribed to the publishing service. Furthermore, a user has to specify the duration of his/her subscription to the portions. In particular, a user can specify different subscription periods for different portions. To support this feature, we introduce the definition of component.

Definition 2. (Component) *Let S be an XML Source. A component over S is a tuple $\langle \text{portion}, \text{start}, \text{end} \rangle$, where $\text{portion} \in \text{Portions}$ is a pre-defined portion presented by the publishing service to a user during the subscription phase or a portion explicitly defined by a user, start is the starting date of the subscription to portion and end is the ending date of the subscription to portion .*

In the following, given a tuple $t = \langle c_1, \dots, c_n \rangle$, we use notation $t.c_i$ to denote the value of component c_i , $i = 1, \dots, n$. Thus for instance, given a component $c \in \text{Components}$, $c.\text{end}$ denotes the ending date of the subscription period to c . As for portions, a user can specify by his/her own the components or can select them from a set of pre-defined components generated by the publishing service. An example of component generated by the publishing service is an annual or semestral subscription to the Times newspaper. The concept of component is introduced in our framework to efficiently manage package delivery to subscribed users. In the following we denote with *Components* the set of all the components defined over the XML Source. Moreover we define a *non expired component*, as a component whose ending date is not yet expired. Finally, we define a package.

Definition 3. (User Package) *Let S be an XML Source. A package over S is a tuple $\langle \text{user_id}, \text{comp}, \text{mode} \rangle$, where user_id is the identifier of a user subscribed to S , $\text{comp} \subseteq \text{Components}$ is a set of components specified by user_id , and mode is the subscription mode, which can be Pull, Notify, Push_{on} and Push_{off} .*

Example 2. Consider four users U_1, U_2, U_3 and U_4 , the XML document in Figure 1(a) and the DTD in Figure 1(b). Suppose that user U_1 is interested in receiving the Times Frontpage in the period from 1/01/01 to 12/31/01, and all the newspaper articles whose title contains the phrase “XML security” in the period from 01/01/01 to 06/30/01. Moreover, suppose that U_1 specifies, during the subscription phase, the Push_{on} mode for all these portions. Suppose that user U_2 is interested in receiving the Frontpage of all the newspapers belonging to the

XML Source of the publishing service in the period from 03/01/01 to 12/31/01. Moreover he wants to receive in the period from 03/01/01 to 12/31/01 all the articles written by Tom Red. Furthermore, U_2 specifies the Pull mode for all those portions. Suppose that user U_3 is interested in receiving the Times **Frontpage** in the period from 01/01/01 to 06/30/01, and that for this portion he specifies the *Push_{on}* mode. Finally, suppose that user U_4 is interested in receiving the Times **Frontpage** in the period from 01/01/01 to 02/28/01, and that for this portion he specifies the *Push_{on}* mode. Thus the portions generated during the subscription phase are the following:

- P_1 : the **Frontpage** element of the instances of the DTD in Figure 1(b);
- P_2 : the **Frontpage** element of the XML document in Figure 1(a);
- P_3 : the **Article** elements of all the instances of the DTD in Figure 1(b), where the attribute **Title** contains the phrase “XML security”;
- P_4 : the **Article** element of all the instances of the DTD in Figure 1(b), where the attribute **Author** has value “Tom Red”.

As a consequence of the different subscription periods selected by the users and of the different subscription modes specified, the packages generated by the publishing service, are the following:

- $PA_1 = \langle U_1, \langle C_2, C_3 \rangle, Push_{on} \rangle$
- $PA_2 = \langle U_2, \langle C_1, C_4 \rangle, Pull \rangle$
- $PA_3 = \langle U_3, \langle C_5 \rangle, Push_{on} \rangle$
- $PA_4 = \langle U_4, \langle C_6 \rangle, Push_{on} \rangle$

where the components are:

- $C_1 = \langle \langle \text{“//Newspaper/Frontpage//node()”}, 03/01/01, 12/31/01 \rangle \rangle$
- $C_2 = \langle \langle \text{“//Newspaper[@Title=“Times”]/Frontpage/node()”}, 01/01/01, 12/31/01 \rangle \rangle$
- $C_3 = \langle \langle \text{“//Newspaper//Article[@Title=“XML security”]/node()”}, 01/01/01, 06/30/01 \rangle \rangle$
- $C_4 = \langle \langle \text{“//Newspaper//Article[@Author=“Tom Red”]/node()”}, 03/01/01, 06/30/01 \rangle \rangle$
- $C_5 = \langle \langle \text{“//Newspaper[@Title=“Times”]/Frontpage/node()”}, 01/01/01, 06/30/01 \rangle \rangle$
- $C_6 = \langle \langle \text{“//Newspaper[@Title=“Times”]/Frontpage/node()”}, 01/01/01, 02/28/01 \rangle \rangle$.

5 Key Distribution

The publishing service, in order to ensure a secure delivery of packages, encrypts all the packages (or portions of them) with proper encryption keys. In this section we illustrate the strategy for keys generation. Two kinds of encryption keys are used:

- *Package Keys*. When a user subscribes to a package in Pull or Notify mode, then the publishing service generates a unique (symmetric) key associated only with this package. Then, different packages with Pull or Notify mode have different package keys.

- *Component Keys.* The encryption for packages in Push mode is based on session keys. Thus, the publishing service generates a different key for each component defined over the source and contained into a package to be delivered according to the Push mode. This means that the same portion with different subscription periods¹ has different associated component keys.

When a user subscribes to a package in Pull or Notify mode, the publishing service sends him/her the corresponding package keys. By contrast, when a user subscribes to a package in Push mode, the publishing service sends him/her the set of keys associated with the components belonging to the package.

Example 3. Consider the packages of Example 2. The publishing service generates, during the subscription phase, the component keys represented in Table 1. Since U_2 subscribed to package PA_2 in the Pull mode, the publishing service generates also the Package key K_{PA_2} .

Table 1. Component keys generated by the publishing service for the packages in Example 2.

Component		Key
Portion	Period	
P_1	03/01/01, 12/31/01	K_{C_1}
P_2	01/01/01, 12/31/01	K_{C_2}
P_2	01/01/01, 06/30/01	K_{C_5}
P_2	01/01/01, 02/28/01	K_{C_6}
P_3	01/01/01, 06/30/01	K_{C_3}
P_4	03/01/01, 06/30/01	K_{C_4}

At the end of the subscription phase, user U_1 receives from the publishing service a key for component C_2 and a key for component C_3 . By contrast user U_2 receives a key for the whole package PA_2 , since the package has to be delivered to U_2 according to the pull distribution mode. Finally, user U_3 receives a key for component C_5 , whereas U_4 receives a key for component C_6 . The keys associated by the publisher service with each user are summarized in Table 2.

6 Pull and Notify Package Distribution Modes

As we have seen in Section 3, if a user subscribes to a package pa in the notify mode, each time a portion of a non expired component of pa is modified, the user receives a notification (i.e., e-mail or message), that advises him/her of the modification. Then, the user can decide if he/she is interested in receiving the modified portion(s) or not. If the user is interested in receiving the modified

¹ Two subscription periods are different if the ending date and/or the starting date are different from each other.

Table 2. Keys associated by the publishing service to users U_1 , U_2 , U_3 and U_4

User	Keys
U_1	$K_{C_2} K_{C_3}$
U_2	K_{PA_2}
U_3	K_{C_5}
U_4	K_{C_6}

portion(s), the user has to explicitly request them to the publishing service as in the case of packages in pull mode. Thus, we can uniformly treat the distribution of packages with notify and pull mode. In both cases, package distribution is based on symmetric encryption. When a user subscribes to a package pa in notify or pull mode, the publishing service returns him/her a package key K_{pa} . Then, when a user requests a package, the publishing service checks which package portions have been modified from the previous request by the same user, encrypts them with the package key K_{pa} , and sends them to the user. The portions that need to be returned to the user are referred to as *Pull Answer Set*, formally defined as follows.

Definition 4. (Pull Answer Set). *Let S be an XML Source, u_id be the identifier of a user submitting an package request, pa be the requested package, and K_{pa} be the key associated with pa . Moreover let $\text{Portion_list} \in \text{Portions}$ be the set of portions which have been modified from the last request of pa by user u_id , and which are contained in non expired components of the package pa . The Pull Answer Set of package pa for user u_id , denoted as $AS_{\text{Pull}}(u_id)$, is the encryption of Portion_list with K_{pa} .*

An Algorithm for generating the Pull Answer Set for a specific user is presented in Figure 3

The Pull Answer Set of a package pa , for a given user, contains all the portions that have been modified from the latest request of the same package by the same user, and that are contained in components whose subscription period has not yet expired. Therefore, we suppose that the publishing service maintains for each portion the date of the latest modification and for each package the date of the latest access by users entitled to access its contents. First, Algorithm 1 checks if the user submitting the request is entitled to access the requested package (step 3). If this is not the case, it returns a *request denied* message. Otherwise, step 4 of the algorithm considers each component belonging to the requested package pa . In step 4.a the algorithm checks if the subscription period of the component has started and not yet expired. Step 4.a.i checks for all the portions of the component, if latest modification has been made after the date of the latest request by the user. If this is the case, it adds the associated portion to set $\text{Portions}_{\text{changed}}$. Then, if $\text{Portions}_{\text{changed}}$ is not empty in step 5.a-b the algorithm encrypts the portions belonging to $\text{Portions}_{\text{changed}}$ with the package

Algorithm 1 *Pull Answer Set generation Algorithm*

```

INPUT:  1  $u\_id$ , the identifier of the user submitting the package request
        2  $pa$ , the requested package
OUTPUT: 1  $AS_{Pull}(u\_id)$  if the request is granted
        2 request denied otherwise

1  $Portions_{changed}$  is initialized to be empty
2 Let Last_req be the date of the latest request of  $pa$  submitted by user  $u\_id$ 
3 If  $pa.user\_id \neq u\_id$ :    Return request denied
4 For each  $c \in pa.comp$ 
    a If  $c.start \leq \text{current-date}$  AND  $c.end \geq \text{current-date}$ :
        i For each  $p \in c.portions$ :
            A Let Last_mod( $p$ ) be the date of the latest update of portion  $p$ 
            B If Last_mod( $p$ ) > Last_req:    Add  $p$  to  $Portions_{changed}$ 
Endfor
5 If  $Portions_{changed} = 0$ 
    Return request denied
else
    a Let  $K_{pa}$  be the key associated with package  $pa$ 
    b  $AS_{Pull}(u\_id) = [Portions_{changed}]_{K_{pa}}$ 
    c Return  $AS_{Pull}(u\_id)$ 
Endif

```

Fig. 3. Pull Answer Set generation Algorithm

key associated with pa , and returns the encrypted portions to the user submitting the package request.

7 Push Package Distribution Mode

Under the Push mode, secure delivery is ensured through the use of session keys. In particular, the difference between $Push_{on}$ and $Push_{off}$ mode is only on the event that triggers the delivery of a package or of some of its portions. In both cases, the system checks whether the portions, which have been modified, are contained in a non expired component. The time of this check depends on the distribution mode. In case of $Push_{on}$, this check is executed upon each portion update, while in $Push_{off}$ this check is executed periodically. Then, the system verifies if these components belong to a package with $Push_{on}$ or $Push_{off}$ mode. In this case, the modified portion(s) need to be returned to all the users subscribed to a package, to which the portion(s) belongs to, together with the keys necessary for their decryption. The portion(s) and the keys are collected into the *Push Answer Set*. For $Push_{on}$ mode the Push Answer Set is formally defined as follows:

Definition 5. (*Push Answer Set*). Let S be an XML Source. Let $p \in \text{Portions}$ be a portion whose content has been modified. Let $\text{Component_list} \subseteq \text{Components}$ be the set of non expired components containing p and belonging to packages whose distribution mode is Push_{on} . Moreover, let U be the set of identifiers of users whose packages contain an element of Component_list . The Push Answer Set of portion p is a tuple $\langle E_{K_s}[p], \text{Encrypted_key} \rangle$, where $E_{K_s}[p]$ is the encryption of p with a session key K_s ; Encrypted_key is a set containing, for each component $c_j \in \text{Component_list}$, the value $E_{K_{c_j}}[K_s]$, that is, the encryption of K_s with the component key of c_j .

Thus, for each modified portion p with a valid subscription period, the Push Answer Set contains the encryption of p with a session key. Moreover, for each non expired component to which p belongs to, the Push Answer Set contains the encryption of the session key with the key associated with the component. Thus, when a user receives a Push Answer Set, he/she can decrypt only those portions belonging to a component for which he/she has the key. An algorithm generating the Push Answer Set for Push_{on} mode is presented in Figure 4.

Algorithm 2 *Push Answer Set generation Algorithm for Push_{on} mode*

INPUT: A portion p , whose content has been modified

OUTPUT: AS_{push} , the Push Answer Set of portion p

```

1 Generate a session key  $K_s$ 
2 Add to  $AS_{push}$  the encryption of  $p$  with  $K_s$ 
3 For each  $c \in \text{Components}$ :
    a If  $p \in c.\text{portions}$ :
        i Let  $PA$  be the set of packages containing  $c$ 
        ii For each  $pa \in PA$ :
            I If  $pa.\text{mode} = \text{Push}_{on}$ :
                A If  $c.\text{start} \leq \text{current-date}$  AND  $c.\text{end} \geq \text{current-date}$ :
                    1 Let  $K_c$  be the key associated with component  $c$ 
                    2 Add the encryption of  $K_s$  with  $K_c$  to  $AS_{push}$ 
Endfor
4 Return  $AS_{push}$ 
```

Fig. 4. Push Answer Set generation Algorithm for Push_{on} mode

Algorithm 2 takes as input a modified portion p , and returns the Push Answer Set for that portion. The algorithm consists of two main steps. The first (i.e., step 2) encrypts portion p with a session key. The second step (i.e., step 3) adds to the Push Answer Set the encryptions of the session key, used for the portion encryption, with the keys of all the components that contain the portion. In particular, Algorithm 2 computes the components containing p (step 3.a). Then, for each package containing one of those components, Algorithm 2 checks if the

package has the $Push_{on}$ distribution mode (steps 3.a.ii and 3.a.ii.I). Moreover, it checks if the subscription period of these components is valid, that is, if it has been started and not yet expired (step 3.a.ii.I.A), and, in this case the algorithm generates the encryption of the session key with the component key, and adds it to the Push Answer Set (step 3.a.A.1). Finally, in Step 6, it returns the computed Push Answer Set. In case of $Push_{off}$ mode the definition of Push Answer Set is slightly modified. In this case the Push Answer Set contains all the portions which have been modified from the latest check, together with the necessary keys. The algorithm for the generation of the Push Answer Set for $Push_{off}$ mode is analogous to Algorithm 2, thus we do not report it in the paper.

8 An Example of Package Delivery

Consider the scenario of Example 2. Suppose that on the 03/30/01 a new issue of the Times newspaper is made available at the publishing service, and that this issue contains an article written by Tom Red. Moreover, suppose that this issue does not contain any articles on XML Security. Therefore, the portions defined in Example 2, that have been changed by this update are: P_1, P_2, P_4 . Let us consider P_1 . This portion denotes all the **Frontpage** elements of the instances of the DTD in Figure 1(b). Thus, portion P_1 contains the Frontpage of the new Time issue. When a portion has been modified the publishing service executes Algorithm 2 to create the corresponding Push Answer Set. In particular, Algorithm 2 checks if the modified portion belongs to a non expired component, contained into package having a $Push_{on}$ mode, and then creates the Push Answer Set. Thus, when the publishing service executes Algorithm 2 for portion P_1 , it verifies that P_1 belongs only to a non expired component (i.e., C_1), but this component belongs to package PA_2 , which has a $Pull$ distribution mode. Therefore, it does not generate the Push Answer Set for P_1 . By contrast, when the publishing service executes Algorithm 2 for portion P_2 , it verifies that P_2 belongs to components C_2, C_5 and C_6 . All the packages containing these components, that is, packages PA_1, PA_3 and PA_4 , have a $Push_{on}$ distribution mode. Thus, the Algorithm checks the subscription period of each of these components. It thus verifies that component C_6 is expired (on date 02/28/01), whereas components C_2, C_5 are not. Thus, the Push Answer Set consists of the encryption of P_2 with a session key K_s , the encryption of K_s with K_{C_2} , that is, the key associated with component C_2 , and the encryption of K_s with K_{C_5} , that is, the key associated with component C_5 . The Push Answer set for portion P_2 is illustrated in Figure 5. Then, the system sends the Push Answer Set returned by Algorithm 2 to the appropriate users, that for portion P_2 are users U_1 and U_3 . Note that if the Push Answer is eavesdropped by user U_4 , he/she does not have the keys necessary to decrypt the session key (indeed, U_4 has only key K_{C_6}). Thus, U_4 cannot decrypt portion P_2 , since his/her subscription period to P_2 is expired.

Finally, when the publishing service executes Algorithm 2 for portion P_4 , it verifies that P_4 belongs to a non expired component (i.e., C_4), and that C_4

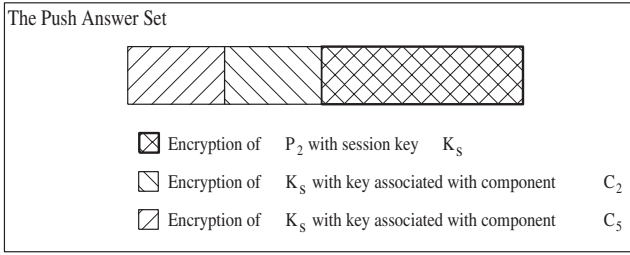


Fig. 5. The Push Answer set for portion P_2

is contained into package PA_2 , whose distribution mode is *Pull*. Therefore, Algorithm 2 does not create the Push Answer Set for P_4 . Let us suppose that after a week, user U_2 requests package PA_2 . In this case, the publishing service executes Algorithm 1, which takes as input the user identifier and the requested package. First, the Algorithm identifies the portions of PA_2 belonging to non expired components, that is, C_1 and C_4 . Then, it verifies for each portions of those components (i.e., P_1 and P_4) if it has been changed from the last request by U_2 . Suppose that the last request has been done in date 03/29/01, therefore all the portion contents are new for user U_2 . Thus, the algorithm creates the Pull Answer Set for U_2 , by encrypting portions P_1 and P_4 with the key associated with package PA_2 by the publishing service during the subscription phase. Then, the publishing service takes the Pull Answer Set returned by Algorithm 2 and sends it to U_2 .

9 Related Work

The problem of securing XML is receiving a growing attention. Securing XML documents entails addressing several issues ranging from access control mechanisms, security policy design, XML encryption, authentication frameworks, secure publishing. An overview of research work and commercial products related to XML security can be found at [5]. However, most of the work reported in [5] focus mainly on access control mechanisms tailored to the protection of XML documents or on the design of encryption techniques for XML documents. No approaches dealing with secure publishing of XML documents have been proposed. Our work is, however, somehow related to work carried out in the context of broadcast applications [4,6,10], where the transmission needs to be encrypted (such as direct broadcast digital TV network or Internet multicasts [7]). Indeed, the *Push* distribution mode proposed as part of our publishing service can be considered as a service for the direct broadcast of document(s) (or portion(s) of them) belonging to an XML document source. In this area the goals are capabilities for flexible content specifications, that is the possibility for users to define personalized packages of documents (or programs), and a secure delivery

of these packages. The publishing service proposed in this paper achieves both of them, whereas among all the approaches presented in [4,6,10] the only ones which offer flexible package specifications are those proposed in [4] and in [6]. However such techniques rely on public-key cryptography [8]. By contrast, the encryption schemes we propose for push distribution use symmetric encryptions. Moreover we complement the push distribution mode with other two modes, i.e., pull and notify, thus achieving a higher degree of flexibility.

10 Concluding Remarks

In this paper we have presented a secure publishing service for XML documents. The service provides the subscribers with a flexible way of specifying the information they are interested in receiving. Additionally, users can subscribe to different portions of the source with different subscription periods, and can select among three different distribution modes – Pull, Push, and Notify, according to which they want to receive information from the service. Secure delivery is ensured through the use of different encryption techniques for the different distribution modes aiming at reducing the number of keys to be generated by guaranteeing at the same time a secure delivery of information to subscribed users.

The work reported in this paper can be extended along several directions. A first direction is the integration of the publishing service we have proposed into Author-*X* [3] – an XML document server providing a comprehensive environment for securing XML documents. In such a way release of information to subscribed users can be regulated not only by the fees the users have paid, but also by the security policies in place at the publishing service. For instance, one of such policies can specify that some of the materials released by the publishing service can be accessed only by users which are greater than eighteen years old. Moreover, we plan to expand the publishing service to ensure authenticity and confidentiality requirements. In particular, we plan to introduce the publishing service into the scalable architecture presented in [1], which is based on a distinction between the Owner and the Publisher of information. In this architecture, by means of a set of digital signatures generated by the Owner and no trust required of the Publisher, a user is able to verify the authenticity of a query answer. A further extension is the definition of an XML-based language for specifying components during the subscription phase, and for guaranteeing the consistency between all the components defined over the XML Source.

References

1. E. Bertino, B. Carminati, E. Ferrari, B. Thuraisingham, A. Gupta. Selective and Authentic Third-party Distribution of XML Document. Technical Report DSI, University of Milano. Submitted for publication.
2. E. Bertino, S. Castano, E. Ferrari. Securing XML Documents: the Author-*X* Project Demonstration. In Proc. of the *ACM SIGMOD 2001* Conference, Santa Barbara, CA, May 2001.

3. E. Bertino, S. Castano, E. Ferrari and M. Mesiti. Specifying and Enforcing Access Control Policies for XML Document Sources. *World Wide Web Journal*, Baltzer Science Publishers, 3(3), 2000.
4. G.C. Chick, S.E. Tavares. Flexible Access Control with Master Keys. *In Proc. of the Conference on Advances in Cryptology (EUROCRYPT '89)*, pages 316-322, 1998.
5. C. Geuer Pollmann. The XML Security Page.
http://www.nue.et-inf.uni-siegen.de/~geuer-pollmann/xml_security.html
6. S. Halevi, and E. Petrank. Storing classified file. Available at
<ftp://theory.lcs.mit.edu/pub/people/shaih/classify.ps.gz>
7. M. Moyer, J.R. Rao, P. Rohatgi. A Survey of Security Issues in Multicast Communications. *IEEE Network* 13, 6(Nov/Dec), 16-23, 1999.
8. RSA Data Security Inc. <http://www.rsa.com>
9. W. Stallings. Network Security Essentials: Applications and Standards. Prentice Hall, 2000.
10. A. Wool. Key Management for Encrypted Broadcast. *ACM Transactions on Information and System Security*, Vol. 3, 107-134, May 2000.
11. Word Wide Web Consortium. Extensible Markup Language (XML) 1.0, 1998.
12. World Wide Web Consortium (1999). XML Path Language (XPath) 1.0.
<http://www.w3.org/TR/xpath>.

An Optimistic Non-repudiation Protocol with Transparent Trusted Third Party

Olivier Markowitch and Steve Kremer

Université Libre de Bruxelles
Département d'Informatique
CP 212 Boulevard du Triomphe
1050 Bruxelles, Belgium
{omarkow,skremer}@ulb.ac.be

Abstract. In this paper we consider a new and efficient optimistic non-repudiation protocol. In a non-repudiation protocol, during which Alice wants to transmit a message to Bob, Alice has to send a non-repudiation of origin evidence to Bob (attesting that Alice is at the origin of the transmitted message), and Bob has to send a non-repudiation of receipt evidence to Alice (attesting Bob's receipt of the message). Classical solutions propose to use a trusted third party to help realizing the exchange without giving any significant advantage to one of the two parties. In an optimistic protocol, the trusted third party intervenes only in case of problems during the communication between Alice and Bob. Classically, in a situation where an error occurs, evidences that have been digitally signed by the TTP are issued. Although these evidences are distinct from those produced by Alice and Bob in a faultless case, they have the same value in case of a dispute. In this paper we propose a protocol where the TTP produces the same evidences that Alice and Bob should have produced in a faultless protocol execution (this prevents, after a succesful protocol execution, to determine whether the TTP was involved or not).

1 Introduction

Due to the overwhelming importance the Internet gained nowadays, more and more sophisticated security services are requested. Non-repudiation services are one of these new security requirements. Non-repudiation services must ensure that when Alice sends some information to Bob over a network, neither Alice nor Bob can deny having participated in a part or the whole of this communication. Therefore a non-repudiation protocol has to generate non-repudiation of origin evidences intended to Bob, and non-repudiation of receipt evidences destined to Alice. In case of a dispute (e.g. Alice denying having sent a given message or Bob denying having received it) an adjudicator can evaluate these evidences and take a decision in favor of one of the parties without any ambiguity. In comparison to other security issues, such as privacy or authenticity of communications, non-repudiation has not been studied intensively. However many applications such as

electronic commerce, fair exchange, certified electronic mail, etc. are related to non-repudiation. Non-repudiation of origin can easily be provided by signing the sent information. A digital signature provides an irrefutable non-repudiation of origin evidence. Non-repudiation of receipt is more difficult to provide: therefore Alice and Bob have to follow a protocol that assures both services. To ensure the correct exchange of the non-repudiation evidences, the protocol has to be fair. Informally, we say that a protocol is fair if at the end of the protocol execution either Alice receives a non-repudiation of receipt evidence and Bob receives a non-repudiation of origin evidence or none of them receives any valid evidence.

First complete protocols providing both non-repudiation of origin and non-repudiation of receipt have been presented by Zhou and Gollmann and by Zhang and Shi [14,16]. Their proposals relies on a trusted third party (TTP) that has to intervene during each protocol run. Such a TTP is said to be online. In Zhou and Gollmann's protocol, the TTP plays the role of a low-weight notary: it only has to publish an evidence in a read-only public directory, accessible to both Alice and Bob. Although the TTP is low-weight it may create a communication bottleneck. Therefore Zhou and Gollmann presented a second protocol [17] based on the optimistic idea introduced for fair exchanges in [2]: he assumes that in general Alice and Bob are honest, i.e. they correctly follow the protocol, and that the TTP only intervenes, by the mean of a recovery protocol, when a problem arises. A TTP that does not necessarily intervene in each protocol run is said to be offline. Zhou makes the assumption that the channels between Alice and Bob are unreliable and that the channels between the two parties and the TTP are resilient, i.e. all messages arrive after a finite, but unknown amount of time. In [10,15] it has been shown that with these assumptions the optimistic protocol does not provide fairness. For the protocol to be fair the channel between Alice and the TTP needs to be operational, i.e. each message arrives after a known, constant amount of time. This is however a very strong and unrealistic assumption. Solutions for optimistic non-repudiation protocols have been proposed in [10,15] where the last one suffers of a little design problem [6].

In this paper we present an optimistic non-repudiation protocol with a *transparent* TTP. In an optimistic protocol, the trusted third party intervenes only in case of problems during the communication between Alice and Bob. Classically, in such a situation the TTP digitally signs some pieces of information which will be used as non-repudiation evidences. These evidences have the same value to an adjudicator than those produced by Alice and Bob in a faultless case. Our aim is to design a protocol, where the TTP is *transparent*. This means that at the end of the protocol, by only looking at the produced evidences, it is impossible to decide whether the TTP did intervene in the protocol execution or not. As the intervention of the TTP can be due to a network failure, rather than a cheating party, transparent TTPs can be very useful in the context of electronic commerce, in order to avoid bad publicity. First related works have been realized by Chen [8], Asokan et al. [3], Bao et al. [4], Boyd and Foo [5] and Markowitch and Saaednia [11]. In the context of purchase of digital goods, Liqn Chen [8] proposed a protocol using discrete logarithm based signatures, in which the

client commits his signature in a verifiable way for the provider. If the client does not send his final signature after having received the item, the TTP transmits information which have the same properties as a client's final signature when combined with the earlier committed signature. However, the signature generated by the TTP in case of a problem, can be distinguished from the signature generated in a faultless case. Hence, the TTP is not transparent. The use of an *invisible TTP* was first proposed by Micali [12] in the framework of certified e-mails. Asokan et al. [3] and Bao et al. [4], proposed fair exchange protocols allowing to recover, in case of problem, the original client's signature committed earlier in the protocol rather than affidavits produced and signed by the TTP. Asokan et al.'s protocol is based on verifiable encryption, which however is computationally inefficient. Bao et al. proposed two protocols, from which the first one is inefficient, while the second one, though more efficient, has recently been broken by Boyd and Foo [5]. In the same paper, Boyd and Foo [5] proposed a fair exchange protocol for electronic payment. Their method allows to recover the original client's signature from the committed one, using designated convertible signatures [7]. They also proposed a concrete protocol based on the RSA signature scheme. However, their scheme requires an additional interactive protocol and hence is rather inefficient. The most efficient protocol for fair exchange with transparent TTP has recently been proposed by Markowitch and Saaednia [11]. The protocol is based on a specific signature scheme (inspired by the Girault-Poupard-Stern signature scheme [13]). It does not need an additional interactive protocol and is efficient considering both communication and computation. All of the here discussed proposals apply to fair exchange protocols. In this paper we aim to present the first non-repudiation protocol using a transparent TTP. Although a non-repudiation protocol could be seen as a special instance of a fair exchange protocol—an exchange of a message and a non-repudiation of origin evidence against a non-repudiation of receipt evidence—there exist several inherent differences. While in a fair exchange protocol, the description of the items to exchange is known a priori, in a non-repudiation protocol the recipient of a message does not expect a particular message (it does not know the description of the message he will obtain at the end of the protocol). Moreover Bob does not exchange an item, but only an evidence of receipt, which is generally required in fair exchanges in addition to the expected item. These differences are rather subtle, but imply more efficient solutions for non-repudiation protocols than instantiations of fair exchange protocols. The protocol, presented in this paper is based on the Markowitch-Saaednia method [11].

The paper is organized as follows. We start giving some basic definitions of the different classes of communication channels considered in this paper, and define the requirements and properties that non-repudiation protocols must respect. We go on presenting an optimistic non-repudiation protocol [10]. Then we present our novel optimistic non-repudiation protocol with transparent TTP, emphasizing the differences with the previous one. Finally we conclude the paper.

2 Basic Definitions and Properties

2.1 Communication Channels

In the framework of such exchange protocols, we can distinguish three classes of communication channels: unreliable channels, resilient channels and operational channels. No assumptions have to be made about unreliable channels: data may be lost. A resilient channel delivers correct data after a finite, but unknown amount of time. Data may be delayed, but will eventually arrive. When using an operational channel correct data arrive after a known, constant amount of time. Operational channels are however rather unrealistic in heterogeneous networks.

2.2 Requirements on Non-repudiation Protocols

In the rest of this paper we suppose that no party acts against its own interests. This assumption is rather natural and avoids us to deal with situations where a dishonest party, i.e. a party not following the protocol, breaks some of the underneath defined properties by adopting a behavior harming itself.

The primary property a non-repudiation protocol between Alice and Bob must provide is *non-repudiability*. To offer complete non-repudiation services, a protocol must provide non-repudiation of receipt and non-repudiation of origin.

Definition 1 (Non-repudiation of receipt). *A non-repudiation protocol provides non-repudiation of receipt, if and only if it generates a non-repudiation of receipt evidence, destined to Alice, that can be presented to an adjudicator, who can unambiguously decide whether Bob received a given message or not.*

Definition 2 (Non-repudiation of origin). *A non-repudiation protocol provides non-repudiation of origin, if and only if it generates a non-repudiation of origin evidence, destined to Bob, that can be presented to an adjudicator, who can unambiguously decide whether Alice is the author of a given message or not.*

However, the non-repudiability property is not sufficient in most cases. For a non-repudiation protocol to be practical we require that none of the involved parties will have an advantage at any given moment. The property reflecting this requirement is *fairness*. Fairness comes in three flavors: *weak* fairness, *strong* fairness and *true* fairness. Weak fairness has been introduced by Asokan et al. [1]. It guarantees that if Bob received his expected item, Alice will either receive her item or an evidence showing that Bob can have access to Alice's item. We will not enter into details of this property as it is not of great importance in non-repudiation protocols.

Definition 3 (Strong fairness). *A non-repudiation protocol provides strong fairness if and only if at the end of a protocol execution either Alice got the non-repudiation of receipt evidence for the message m , and Bob got the corresponding*

message m as well as the non-repudiation of origin evidence for this message, or none of them got any valuable information.

Note that the non-repudiation of receipt and/or origin could consist of an evidence generated by the TTP.

Definition 4 (True fairness). *A non-repudiation protocol provides true fairness if and only if it provides strong fairness and, if the exchange is successful, the non-repudiation evidences produced during the protocol are independent of how the protocol is executed.*

When a non-repudiation protocol respects the true fairness property, it is impossible, by looking only at the evidences, to decide whether the TTP did intervene or not in the protocol. The TTP, if solicited, has to produce evidences indistinguishable from those issued by Alice and Bob in faultless cases. Such a TTP is transparent.

Timeliness is another property, needed in order for a protocol to be practical.

Definition 5 (Timeliness). *A non-repudiation protocol provides timeliness if and only if all honest parties always have the ability to reach, in a finite amount of time, a point in the protocol where they can stop the protocol while preserving fairness.*

Timeliness avoids situations where a party does not know whether it can stop the protocol without losing fairness or not.

3 A Non-repudiation Protocol with Offline TTP

3.1 Introduction

We will now present a two-party non-repudiation protocol with an off-line TTP [10]. In this protocol, Alice wants to exchange a message m and its corresponding non-repudiation of origin evidence against a non-repudiation of receipt evidence, issued by Bob. This protocol results from modifications made on the Zhou-Gollmann optimistic protocol [17] in which an operational channel is needed between the TTP and Alice in order to assure fairness. The here presented protocol only needs a resilient channel between the TTP and respectively Alice and Bob. The channel between Alice and Bob may even be unreliable. The protocol is similar to the independently developed autonomous two-party non-repudiation protocol proposed earlier by Zhou et al. in [15]. However Zhou et al.'s protocol suffers from a small design error, pointed out in [6].

The protocol is composed of three sub-protocols: the main protocol, the recovery protocol and the abort protocol. The main protocol consists of messages exchanged directly between Alice and Bob. In case of problems during this main

protocol, two (mutually exclusive) possibilities are offered to the entities. Either Alice contacts the TTP to abort the protocol in order to cancel the exchange, or Alice or Bob contacts the TTP to launch the recovery protocol in order to complete the exchange.

3.2 Notations and Evidences

We use the following notation to describe the protocol.

- $X \rightarrow Y$: transmission from entity X to entity Y
- $h()$: a collision resistant one-way hash function
- $E_k()$: a symmetric-key encryption function under key k
- $D_k()$: a symmetric-key decryption function under key k
- $E_X()$: a public-key encryption function under X 's public key
- $D_X()$: a public-key decryption function under X 's private key
- $S_X()$: the signature function of entity X
- m : the message sent from A to B
- k : the session key A uses to cipher m
- $c = E_k(m)$: the cipher of m under the session key k
- $l = h(m, k)$: a label that in conjunction with (A, B) uniquely identifies a protocol run
- f : a flag indicating the purpose of a message

During the protocol the following evidences are generated.

- the evidence of origin for the cipher c : $\text{EOO} = S_A(f_{\text{EOO}}, B, TTP, l, h(c))$
- the evidence of receipt for the cipher c : $\text{EOR} = S_B(f_{\text{EOR}}, A, TTP, l, h(c))$
- the evidence of submission of key k : $\text{Sub} = S_A(f_{\text{Sub}}, B, l, E_{TTP}(k))$
- the evidence of origin for key k : $\text{EOO}_k = S_A(f_{\text{EOO}_k}, B, l, k)$
- the evidence of receipt for key k : $\text{EOR}_k = S_B(f_{\text{EOR}_k}, A, l, k)$
- the recovery request: $\text{Rec}_X = S_X(f_{\text{Rec}_X}, Y, l)$
- the confirmation evidence for key k : $\text{Con}_k = S_{TTP}(f_{\text{Con}_k}, A, B, l, k)$
- the abort request: $\text{Abort} = S_A(f_{\text{Abort}}, B, l)$
- the abort confirmation: $\text{Con}_a = S_{TTP}(f_{\text{Con}_a}, A, B, l)$

3.3 Main Protocol

The basic idea of the main protocol is to first exchange the cipher of the message m against a receipt for this cipher. Secondly, we exchange the decryption key against a receipt for this key. Each transmission is associated to some maximum time-out. Once this time-out value is exceeded the recipient supposes that the transmission will not arrive any more and initiates either a recovery or an abort protocol.

1. $A \rightarrow B$: $f_{\text{EOO}}, f_{\text{Sub}}, B, TTP, l, c, E_{TTP}(k), \text{EOO}, \text{Sub}$
2. $B \rightarrow A$: $f_{\text{EOR}}, A, TTP, l, \text{EOR}$
- if A times out then abort
3. $A \rightarrow B$: $f_{\text{EOO}_k}, B, l, k, \text{EOO}_k$
- if B times out then recovery[$X := B, Y := A$]
4. $B \rightarrow A$: $f_{\text{EOR}_k}, A, l, \text{EOR}_k$
- if A times out then recovery[$X := A, Y := B$]

Alice starts the protocol by sending the cipher of the message, as well as the decryption key, ciphered under the public key of the TTP, to Bob. The message does also contain Alice's signature on the encrypted key and the hash of the cipher. These signatures serve as evidences of origin for the ciphers. If Bob receives the first message he replies with a receipt to confirm the arrival of the first message. This receipt contains Bob's signature on the hash of the cipher c and serves to Alice as an evidence of receipt for the cipher. Alice goes on sending to Bob the decryption key k , as well as her signature on this key. This signature is used as an evidence of origin for the key. The evidence of origin for the cipher c , together with the evidence of origin for the key k , form together the non-repudiation of origin evidence of the message m . Bob replies with a receipt for the key to Alice: his signature on the key k . The signature serves as the evidence of receipt for the key. Together with the evidence of receipt for the cipher c , they form the non-repudiation of receipt evidence of the message m . If at some moment in the protocol, one of the messages does not arrive in a reasonable amount of time, i.e. before the local time-out, Alice or Bob execute an abort or a recovery protocol as indicated in the protocol description. For a more detailed description of this protocol, we refer the reader to [10].

3.4 Recovery Protocol

To launch the recovery protocol Alice or Bob has to send to the TTP several evidences that prove to the TTP that Alice sent the cipher c to Bob and that Bob really received it, i.e. that the protocol has been started between Alice and Bob. Note that the recovery protocol can only be executed once per protocol run and is mutually exclusive with the abort protocol.

1. $X \rightarrow TTP$: $f_{\text{Rec}_X}, f_{\text{Sub}}, Y, l, h(c), E_{TTP}(k), \text{Rec}_X, \text{Sub}, \text{EOR}, \text{EOO}$
if *aborted or recovered* then stop
else *recovered*=true
2. $TTP \rightarrow A$: $f_{\text{Con}_k}, A, B, l, k, \text{Con}_k, \text{EOR}$
3. $TTP \rightarrow B$: $f_{\text{Con}_k}, A, B, l, k, \text{Con}_k$

When the first message arrives, the TTP checks whether an abort protocol or a recovery protocol has already been accepted for this protocol run: a protocol run is uniquely identified by the label $l = h(m, k)$ and the identities (A, B) . If either an abort or a recovery protocol has already been initiated the TTP halts. If the

TTP accepts to perform a recovery protocol, the TTP sends to Alice as well as to Bob all possibly missing evidences. If the recovery protocol is executed, the key confirmation evidence Con_k signed by the TTP will make part of the non-repudiation evidences for the message m . It is used to replace both the evidence of origin for the key as well as the evidence of receipt for the key.

3.5 Abort Protocol

Alice has the possibility to run an abort protocol. If she decides to do so she sends a signed abort request, including label l , to the TTP. If the TTP accepts the request (neither a recovery nor an abort has yet been initiated), the TTP sends to both Alice and Bob a signed abort confirmation.

1. $A \rightarrow TTP : f_{\text{Abort}}, l, B, \text{Abort}$
if *aborted* or *recovered* then stop
else *aborted*=true
2. $TTP \rightarrow A : f_{\text{Con}_a}, A, B, l, \text{Con}_a$
3. $TTP \rightarrow B : f_{\text{Con}_a}, A, B, l, \text{Con}_a$

3.6 Dispute Resolution

The non-repudiation of origin and receipt evidences for message m are the following:

- $\text{NRO} = (\text{EOO}, \text{EOO}_k)$ or $\text{NRO} = (\text{EOO}, \text{Con}_k)$
- $\text{NRR} = (\text{EOR}, \text{EOR}_k)$ or $\text{NRR} = (\text{EOR}, \text{Con}_k)$

Repudiation of origin. When Alice denies the origin of the message, Bob has to present to the judge EOO , EOO_k or Con_k , TTP , l , c , m and k . The judge verifies that:

- $\text{EOO} = S_A(f_{\text{EOO}}, B, TTP, l, c)$,
- $\text{EOO}_k = S_A(f_{\text{EOO}_k}, B, TTP, l, k)$ or $\text{Con}_k = S_{TTP}(f_{\text{Con}_k}, A, B, l, k)$,
- $l = h(m, k)$,
- $c = E_k(m)$.

If Bob can provide all the required items and all the checks hold, the adjudicator claims that Alice is at the origin of the message.

Repudiation of receipt. When Bob denies receipt of m , Alice can prove his receipt of the message by presenting EOR , EOR_k or Con_k , TTP , l , c , m and k to a judge. The judge verifies that:

- $\text{EOR} = S_B(f_{\text{EOR}}, A, TTP, l, h(c)),$
- $\text{EOR}_k = S_B(f_{\text{EOR}_k}, A, l, k)$ or $\text{Con}_k = S_{TTP}(f_{\text{Con}_k}, A, B, l, k),$
- $l = h(m, k),$
- $c = E_k(m).$

If Alice can present all of the items and all the checks hold, the adjudicator concludes that Bob received the message.

3.7 Fairness and Timeliness

We do not discuss fairness and timeliness in detail, but will only give a short idea, why the protocol does provide those properties. A more complete discussion can be found in [10]. If Bob stops the protocol after having received the first message, Alice may perform the abort protocol, in order to avoid that Bob initiates a recovery later. As neither Bob nor Alice received complete evidences the protocol remains fair. If Bob had already initiated the recovery protocol (in that case the abort request is refused by the TTP), the TTP sends all the missing evidences to Alice and Bob. Note that the TTP also sends the EOR to Alice, as she has not received it yet. Thus the protocol stays fair. Once the second message of the main protocol has been received by Alice, both Alice and Bob can launch a recovery protocol and force the successful exchange of the message and the non-repudiation evidences. Note that the protocol achieves strong fairness, but not true fairness. At the end of the protocol, when looking at the evidences, we can easily decide whether the TTP did intervene or not. In the case the TTP intervened in the protocol, the evidences EOR_k and possibly EOO_k have been substituted by the evidence Con_k , generated by the TTP.

When looking at the timeliness, three situations may arrive: the main protocol ends up successfully (without any time-out); Alice aborts the protocol and the abort confirmation signed by the TTP arrives to both Alice and Bob after a finite amount of time, as the channels between the TTP and both Alice and Bob are resilient; a recovery protocol is performed and Alice and Bob receive the evidences after a finite amount of time because of the resilience of the channels.

4 A Non-repudiation Protocol with Offline Transparent TTP

We will now describe the variant protocol where the offline TTP produces, when a fault occurs during the main protocol execution, exactly the same evidences than those produced by Alice and Bob in a faultless case. The protocol described underneath supposes a resilient channel between the TTP and Alice and between the TTP and Bob. The communication channel between Alice and Bob may be unreliable.

4.1 The Signature Scheme

The protocol uses a signature scheme based on the GPS signature scheme [9,13]. The GPS signature of a message m is realized on one hand by choosing a random value r and computing $t = \alpha^r \bmod n$ where n is a composite modulus and α is a basis of order $\lambda(n)$, and on the other hand by computing $z = r + x \cdot h(t, m)$ where x is a secret value associated to $y \equiv \alpha^{-x} \bmod n$ the corresponding public value. The verification is achieved by comparing t and $\alpha^z \cdot y^{h(t, m)} \bmod n$.

The signature used in this protocol is issued in two phases. First the signer produces a committed signature. Then this committed signature is turned into a final signature either by the signer or by the TTP. The recipient of a committed signature is able to check whether the TTP has the ability to transform the committed signature into the signer's final signature.

During an initialization phase, the TTP chooses an integer $n = pq$, where p and q are large random strong primes (of almost the same size). The TTP chooses also a base α of order $\lambda(n)$ and a small integer c such that $\gcd(\lambda(n), c) = 1$. The TTP computes d such that $cd \equiv 1 \pmod{\lambda(n)}$ and $\beta = \alpha^c \bmod n$. Finally, the TTP makes n , β , c and α public, keeps d secret and discards p and q .

A signer u chooses a random integer x_u as secret key and computes the relative public key $y_u = \alpha^{x_u} \bmod n$.

To produce the committed signature on a message m the signer u chooses a random r_u and computes $t_u = \beta^{r_u} \bmod n$ and $z_u = c \cdot r_u + h(t_u, m) \cdot x_u$. The pair (t_u, z_u) forms the committed signature of signer u and will also be noted $\text{ComSig}_u(m)$.

A verifier v can check the committed signature by comparing $\alpha^{z_u} \bmod n$ and $t_u \cdot y_u^{h(t_u, m)} \bmod n$.

The final signature of signer u can be computed independently by the signer u by computing $t'_u = \alpha^{r_u} \bmod n$, or by the TTP by computing $t'_u = t_u^d \bmod n$. The pair (t'_u, z_u) forms the final signature of signer u and will also be noted $\text{FinalSig}_u(m)$.

The verifier checks the validity of the final signature by comparing $\alpha^{z_u} \bmod n$ to $t'^c_u \cdot y_u^{h(t'^c_u \bmod n, m)} \bmod n$ (in practice, it is sufficient to verify that $t = t'^c_u \bmod n$).

The security of this signature scheme has been studied in [11,13].

4.2 Evidences and Notations

The notation used to describe the protocol is the same as in the previous section.

The evidences generated during the protocol are the following.

- the evidence of origin: $\text{EOO} = \text{ComSig}_A(f_{\text{NRO}}, B, \text{TTP}, l, h(c), h(k))$
- the evidence of receipt: $\text{EOR} = \text{ComSig}_B(f_{\text{NRR}}, A, \text{TTP}, l, h(c), h(k))$

- the non-repudiation of origin evidence:
 $\text{NRO} = \text{FinalSig}_A(f_{\text{NRO}}, B, TTP, l, h(c), h(k))$
- the non-repudiation of receipt evidence:
 $\text{NRR} = \text{FinalSig}_B(f_{\text{NRR}}, A, TTP, l, h(c), h(k))$
- the evidence of submission for key k : $\text{Sub} = S_A(f_{\text{Sub}}, B, l, E_{TTP}(k))$
- the abort request: $\text{Abort} = S_A(f_{\text{Abort}}, B, l)$
- the recovery request: $\text{Rec}_X = S_X(f_{\text{Rec}_X}, Y, l)$
- the abort confirmation: $\text{Con}_a = S_{TTP}(f_{\text{Con}_a}, A, B, l)$
- the error confirmation: $\text{Con}_e = S_{TTP}(f_{\text{Con}_e}, A, B, l)$

4.3 Main Protocol

1. $A \rightarrow B$: $f_{\text{EOO}}, f_{\text{Sub}}, B, TTP, l, h(k), c, E_{TTP}(k), \text{EOO}, \text{Sub}$
2. $B \rightarrow A$: $f_{\text{EOR}}, A, TTP, l, \text{EOR}$
 if A times out then abort
3. $A \rightarrow B$: $f_{\text{NRO}}, B, l, k, \text{NRO}$
 if B times out then recovery[$X := B, Y := A$]
4. $B \rightarrow A$: $f_{\text{NRR}}, A, l, \text{NRR}$
 if A times out then recovery[$X := A, Y := B$]

Alice starts the protocol by transmitting to Bob the cipher c of the message m under the session key k , the hash of this session key, the session key ciphered with the TTP's ciphering public key, the committed signature **EOO** (which is the committed non-repudiation of origin evidence) and the evidence of origin of the session key ciphered for the TTP.

Bob verifies the received message and checks the committed signature **EOO** as indicated previously and **Sub**. If the verification holds, Bob sends to Alice his committed signature **EOR** (which is the committed non-repudiation of receipt evidence).

If Alice does not receive the protocol's second message (from Bob) before a local time-out (chosen by herself), or if the received information are incorrect (the message is not well formed or Bob's committed signature is invalid) she realizes the abort protocol described below. Otherwise, she sends to Bob the session key k and the non-repudiation of origin evidence (her final signature **NRO**).

If the information received by Bob are correct (well formed message and valid **NRO** with regard to the session key k he just received), he sends the non-repudiation of receipt evidence (his final signature **NRR**). Otherwise, he initiates the recovery protocol, described underneath

Eventually, if Alice does not receive a correct final sending from Bob she initiates the recovery protocol.

4.4 Recovery Protocol

We here consider that X is the party initiating the recovery, Y being the other party.

1. $X \rightarrow TTP :$ $f_{\text{Rec}_X}, f_{\text{Sub}}, Y, l, h(c), h(k), E_{TTP}(k), \text{Rec}_X, \text{Sub}, \text{EOR}, \text{EOO}$
 if $h(k) \neq h(D_{TTP}(E_{TTP}(k)))$ then error
 if *aborted* or *recovered* then stop
 else *recovered*=true
2. $TTP \rightarrow A :$ $f_{\text{NRR}}, A, l, \text{NRR}$
3. $TTP \rightarrow B :$ $f_{\text{NRO}}, B, l, k, \text{NRO}$

At any time after having received the first message of the main protocol, Bob can initiate the recovery protocol. Alice, after having received the second message of the main protocol, can also initiate the recovery protocol (for example if Bob does not send the fourth message of the main protocol).

The initiator of the recovery protocol sends to the TTP the hash of the ciphered message, the hash of the session key k , the session key ciphered for the TTP and the signatures Rec_X , Sub , EOR and EOO . The TTP first verifies that all the signatures are correct. If at least one signature is incorrect, the request is ignored. These checks also make it impossible for Bob of trying to recover the protocol with a wrong session key k , as the submission evidence Sub has been signed by Alice. Then the TTP verifies that the hash of the key committed in the first message of the main protocol corresponds to the key ciphered under the TTP's public key. If the keys are different, the error protocol described below is launched to inform Bob, that Alice is trying to cheat. Otherwise the TTP checks whether neither the abort nor the recovery protocol have yet been performed. If not, the TTP uses its private key d to convert the committed signatures into final ones. Then the TTP forwards the non-repudiation of receipt evidence (Bob's final signature) to Alice and the non-repudiation of origin evidence (Alice's final signature) to Bob.

4.5 Abort Protocol

1. $A \rightarrow TTP :$ $f_{\text{Abort}}, l, B, \text{abort}$
 if *aborted* or *recovered* then stop
 else *aborted*=true
2. $TTP \rightarrow A :$ $f_{\text{Con}_a}, A, B, l, \text{Con}_a$
3. $TTP \rightarrow B :$ $f_{\text{Con}_a}, A, B, l, \text{Con}_a$

If Alice does not receive the second message of the main protocol, she initiates the abort protocol, by sending an abort request to the TTP. If the protocol have not yet been recovered or aborted, the TTP sends to both Alice and Bob a signed abort confirmation.

4.6 Error Protocol

- aborted*=true
1. $TTP \rightarrow A :$ $f_{\text{Con}_e}, A, B, l, \text{Con}_e$
 2. $TTP \rightarrow B :$ $f_{\text{Con}_e}, A, B, l, \text{Con}_e$

The TTP runs this error protocol if during a recovery protocol it appears that Alice provided a session key to be recovered (thanks to $E_{TTP}(k)$) different from the initially committed session key (the hash of the key is included in EOO¹). The goal of this protocol is to warn Bob that Alice tried to cheat (and to inform Alice that this attempt has been detected detected²).

4.7 Disputes Resolutions

The two final signatures NRO and NRR form respectively the non-repudiation of origin and receipt evidences for message m .

Repudiation of origin. If Alice denies being the author of the message m , Bob presents NRO, TTP , l , c , m and k to an adjudicator. The judge verifies whether the NRO is valid with regard to the session label and the ciphered message provided, $l = h(m, k)$ and $c = E_k(m)$. If Bob provides the required information and all the checks hold, the adjudicator accepts the claim that Alice is at the origin of the message.

Repudiation of receipt. If Bob denies having received the message m , Alice presents to the adjudicator NRR, TTP , l , c , m and k . The judge verifies whether the NRR is valid with regard to the session label and the ciphered message provided, $l = h(m, k)$ and $c = E_k(m)$. If Alice presents all needed information and the checks hold, the adjudicator concludes that Bob received the message.

4.8 Fairness and Timeliness

We start showing that our protocol provides fairness. If Bob stops the protocol after having received the first message, Alice can run the abort protocol to prevent Bob to initiate a recovery later. As neither Bob nor Alice received the non-repudiation evidences (neither NRO nor NRR), the protocol remains fair.

If Bob had already previously initiated the recovery protocol, the TTP forwards to both Alice and Bob all the possibly missing non-repudiation evidences and the protocol stays fair. If Bob is unable to run the recovery protocol, because Alice provided, at the beginning of the main protocol, a session key ciphered for the TTP which differs from the session key hashed (and signed in the EOO), the TTP will launch the error protocol in order to inform Bob that Alice tried to cheat. The protocol will also end in a fair way with no evidences exchanged.

¹ If Bob is the initiator of the recovery protocol, he cannot send a wrong ciphered session key because he has to provide a correct key submission evidence **Sub**, signed by Alice, at the beginning of the recovery protocol.

² Of course, the first message of this error protocol is optional.

If Alice does not send the third message during the main protocol, Alice and Bob may initiate the recovery protocol. Again the protocol will end in a fair way with either all the non-repudiation evidences forwarded to Alice and Bob by the TTP, or with an error message, issued by the error protocol, and no exchanged evidences.

If Alice realizes the third step, Bob receives the non-repudiation of origin evidence. Bob can then send the fourth message of the main protocol and Alice receives the non-repudiation of receipt evidence. If Bob does not send the last message of the main protocol, Alice runs the recovery protocol, and thanks to the resilience of the channels between the TTP and both Alice and Bob, all data sent by the TTP to Alice and Bob eventually arrive. In those cases all entities receive valid evidences and the protocol finishes in a fair way.

Still consider the following scenario, where Alice tries to cheat by signing in the EOO a session key that differs from the one ciphered for the TTP. In that case Alice will never send the third message of the main protocol, in order not to harm herself. Suppose Alice sends the third message and Bob does not reply by sending the fourth message. Alice cannot perform a recovery protocol (since she is unable to provide coherent information to the TTP), and Bob will get his non-repudiation evidence, while Alice does not get her evidence. Such a behavior would contradict the assumption we made in section 2, that says that no entity acts against its own interests. Hence message 3 will not be sent in the main protocol, and the evidences will not be exchanged. Thus the protocol remains fair.

The protocol provides strong and true fairness: when looking at the evidences, no one can determine whether the TTP did intervene or not.

When looking at the timeliness, we have to consider three situations which may arrive: the main protocol ends up successfully (without any time-out); Alice aborts the protocol and the abort confirmation signed by the TTP arrives at Alice and Bob after a finite amount of time, as the channels between the TTP and both Alice and Bob are resilient; a recovery protocol is performed and Alice and Bob receive either the non-repudiation evidences or an error information (via the error protocol) after a finite amount of time because of the resilience of the channels.

5 Conclusion

We have considered a new and efficient fair optimistic non-repudiation protocol. A non-repudiation protocol between Alice and Bob, is such that at the end of the protocol, Alice has sent to Bob a message and a non-repudiation of origin evidence for this message, while Bob has sent to Alice a non-repudiation of receipt evidence for the message. In previous optimistic non-repudiation protocols, the TTP, used during the protocol in case of problem, produces non-repudiation evidence which are distinct from the non-repudiation evidences exchanged by Alice

and Bob in a faultless case. The protocol we propose is the first non-repudiation protocol which features a transparent TTP, i.e. a TTP who produces, in case of problem, the same evidences that Alice and Bob should have produced in a faultless protocol execution. In consequence, at the end of the protocol, by only looking at the exchanged non-repudiation evidences, it is impossible to decide whether the TTP did intervene in the protocol execution or not.

As it is difficult to determine whether the TTP was required during the non-repudiation protocol because of a dishonest party or because of a network problem, such as transparent TTP may be particularly relevant, for example, in an electronic commerce environment.

References

1. N. Asokan. *Fairness in Electronic Commerce*. PhD thesis, University of Waterloo, May 1998.
2. N. Asokan, M. Schunter, and M. Waidner. Optimistic protocols for fair exchange. In T. Matsumoto, editor, *4th ACM Conference on Computer and Communications Security*, pages 6, 8–17, Zurich, Switzerland, Apr. 1997. ACM Press.
3. N. Asokan, V. Shoup, and M. Waidner. Optimistic fair exchange of digital signatures. In *Advances in Cryptology: Proceedings of Eurocrypt'98*, volume 1403 of *Lecture Notes in Computer Science*, pages 591–606. Springer-Verlag, 1998.
4. F. Bao, R. H. Deng, and W. Mao. Efficient and practical fair exchange protocols with off-line TTP. In *IEEE Symposium on Security and Privacy*, May 1998.
5. C. Boyd and E. Foo. Off-line fair payment protocols using convertible signatures. *Lecture Notes in Computer Science*, 1514:271–285, 1998.
6. C. Boyd and P. Kearney. Exploring fair exchange protocols using specification animation. In *The Third International Workshop on Information Security - ISW2000*, *Lecture Notes in Computer Science*, Australia, Dec. 2000. Springer-Verlag.
7. D. Chaum. Designated confirmer signatures. In A. D. Santis, editor, *Advances in Cryptology: Proceedings of Eurocrypt'94*, volume 950 of *Lecture Notes in Computer Science*, pages 86–91. Springer-Verlag, 1995, 9–12 May 1994.
8. L. Chen. Efficient fair exchange with verifiable confirmation of signatures. *Lecture Notes in Computer Science*, 1514:286–299, 1998.
9. M. Girault. Self-certified public keys. In *Advances in Cryptology: Proceedings of EuroCrypt'91*, volume 547 of *Lecture Notes in Computer Science*, pages 490–497. Springer-Verlag, 1991.
10. S. Kremer and O. Markowitch. Optimistic non-repudiable information exchange. In J. Biemond, editor, *21st Symp. on Information Theory in the Benelux*, pages 139–146, Wassenaar (NL), May25-26 2000. Werkgemeenschap Informatie- en Communicatietheorie, Enschede (NL).
11. O. Markowitch and S. Saeednia. Optimistic fair-exchange with transparent signature recovery. In *5th International Conference, Financial Cryptography 2001*, *Lecture Notes in Computer Science*. Springer-Verlag, 2001.
12. S. Micali. Certified E-mail with invisible post offices. Available from author; an invited presentation at the RSA '97 conference, 1997.
13. G. Poupard and J. Stern. Security analysis of a practical “on the fly” authentication and signature generation. In *Advances in Cryptology: Proceedings of Eurocrypt'98*, volume 1403 of *Lecture Notes in Computer Science*, pages 422–436. Springer-Verlag, 1998.

14. N. Zhang and Q. Shi. Achieving non-repudiation of receipt. *The Computer Journal*, 39(10):844–853, 1996.
15. J. Zhou, R. Deng, and F. Bao. Evolution of fair non-repudiation with TTP. In *ACISP: Information Security and Privacy: Australasian Conference*, volume 1587 of *Lecture Notes in Computer Science*, pages 258–269. Springer-Verlag, 1999.
16. J. Zhou and D. Gollmann. A fair non-repudiation protocol. In *IEEE Symposium on Security and Privacy*, Research in Security and Privacy, pages 55–61, Oakland, CA, May 1996. IEEE Computer Society, Technical Committee on Security and Privacy, IEEE Computer Security Press.
17. J. Zhou and D. Gollmann. An efficient non-repudiation protocol. In *PCSFW: Proceedings of The 10th Computer Security Foundations Workshop*. IEEE Computer Society Press, 1997.

Persistent Authenticated Dictionaries and Their Applications^{*}

Aris Anagnostopoulos¹, Michael T. Goodrich², and Roberto Tamassia¹

¹ Dept. Computer Science, Brown University. {aris,rt}@cs.brown.edu

² Dept. Computer Science, University of California, Irvine. goodrich@acm.org

Abstract. We introduce the notion of *persistent authenticated dictionaries*, that is, dictionaries where the user can make queries of the type “was element e in set S at time t ?” and get authenticated answers. Applications include credential and certificate validation checking in the past (as in digital signatures for electronic contracts), digital receipts, and electronic tickets. We present two data structures that can efficiently support an infrastructure for persistent authenticated dictionaries, and we compare their performance.

1 Introduction

At its core, nonrepudiation involves making cryptographically strong statements about the past. Although we digitally sign statements in the present, we only worry about enforcing statements made in the past. Consider, for example, the following scenarios:

- Alice executed a digital mortgage two years ago and now is in default. The bank can only sue Alice if it can prove that she was in fact the one who digitally signed the mortgage contract. (Moreover, the exact history of Alice’s digital certificates may be crucial here, for her signature should remain valid even if her private key was compromised and her digital certificate revoked even just a few weeks after she signed the contract.)
- Bob signed a digital receipt for Alice’s electronic payment and then shipped her a defective item. Alice must be able to prove that Bob truly was the one who signed that receipt.
- A company, CryptoTicket.com, issued a signed digital ticket to Alice for the opera, but Alice is refused entry to the performance. She needs to be able to prove that indeed CryptoTicket.com issued that ticket and was also authorized to do so.
- A company, CryptoNet.com, has published an online catalog that advertised widgets at \$1 a piece on the day they accepted a digital purchase order (PO) for 100 widgets from Alice. Now the company has raised the price to \$100 and is demanding \$10,000 from Alice. She needs to be able to prove that \$1 was the valid price for widgets on the day they accepted her digital PO.

^{*} Research supported in part by DARPA Grant F30602-00-2-0509.

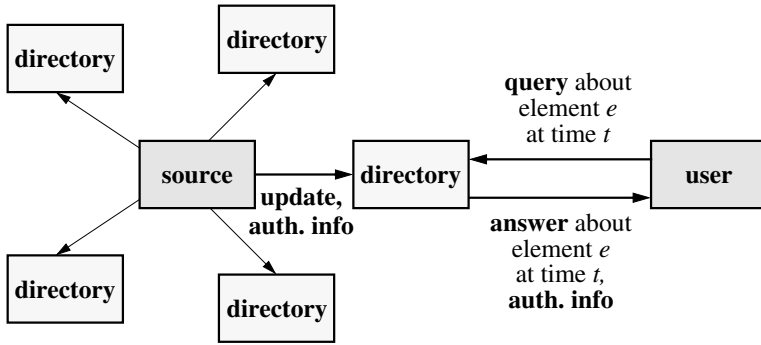


Fig. 1. Persistent authenticated dictionary.

Thus, in order to enforce nonrepudiation on important contractual statements, such as in these examples, we need to have in place a mechanism for checking credentials, certificates, and published information in the past. Ideally, we would like there to be a collection of potentially untrusted directories that can answer historical queries about such items.

1.1 Problem Definition and Applications

Put more abstractly, the problem we address involves three types of parties: a trusted source, untrusted directories, and users. The *source* defines a finite set S of elements that evolves over time through insertions and deletions of elements. Each *directory*, acting as a query agent for the source maintains a copy of set S and receives time-stamped updates from the source together with *update authentication information*, such as signed statements about the update and the current elements of the set. A *user* performs membership queries on the set S of the type “was element e in S at time t ?” but instead of contacting the source directly, it queries one of the directories. The contacted directory provides the user with a yes/no answer to the query together with *query authentication information*, which yields a cryptographic proof of the answer assembled by combining statements signed by the source. The user then verifies the proof by relying solely on its trust in the source and the availability of public information about the source that allows the user to check the source’s signature. We call the data structure used by the directory to maintain the set S , together with the protocol for queries and updates a *persistent authenticated dictionary (PAD)*.

The PAD abstract data type extends the usual notion of an authenticated dictionary [15], where the queries are only of the form “is element e currently in S ?”. Thus, a PAD has the added burden of having to archive the entire history of updates sent by the source to the directories. Moreover, the directories must be able to provide answers and proofs for queries related to any time, past or present. We show, in Fig. 1, a schematic view of a persistent authenticated dictionary.

The design of a persistent authenticated dictionary should address the following goals:

- *Low computational cost:* The computations performed internally by each entity (source, directory, and user) should be simple and fast. More importantly, the space needed to archive historical copies of S should be small.
- *Low communication overhead:* source-to-directory communication (update authentication information) and directory-to-user communication (query authentication information) should be kept as small as possible.
- *High security:* the authenticity of the data provided by a directory should be verifiable with a high degree of certainty.

In addition to the motivating examples given above, applications of persistent authenticated dictionaries include third-party publication of historical data on the Internet [3] and historical certificate revocation checking in public key infrastructures [9,15,11,2,8,5]. In the third-party publication application [3], the source is a trusted organization (e.g., a stock exchange) that produces and maintains historical integrity-critical content (e.g., stock prices) and allows third parties (e.g., Web portals), to publish this content on the Internet so that it becomes widely disseminated. The publishers store copies of the content produced by the source and process queries on such content made by the users.

In the certificate revocation application [9,15,11,2,8,5], the source is a *certification authority* (CA) that digitally signs certificates binding entities to their public keys, thus guaranteeing their validity. Nevertheless, certificates are sometimes revoked (e.g., if a private key is lost or compromised, or if someone loses his authority to use a particular private key). Thus, the user of a certificate must be able to verify that a given certificate used to validate someone for a historic contract was not revoked at the time the contract was executed. To facilitate such queries, *certificate revocation directories* process historic revocation status queries on behalf of users. The results of such queries need to be trustworthy, for they often form the basis for electronic commerce transactions.

In this paper, we present a new scheme for persistent authenticated dictionaries, based on efficient persistent data structures known as red-black trees and skip lists. Our structures are secure, as well as being fast and space efficient in terms of the parameters n , which denotes the current number of elements of the set S , and m , which denotes the total number of updates the source has performed.

1.2 Previous and Related Work

We are not aware of any previous work on persistent authenticated dictionaries. We summarize prior work on *ephemeral* authenticated dictionaries, where only the current version of S is maintained, and on persistent (non-authenticated) dictionaries.

Ephemeral Authenticated Dictionaries. Work has been done on ephemeral authenticated dictionaries primarily in the context of certificate revocation. The

traditional method for certificate revocation (e.g., see [9]) is for the CA (source) to sign a statement consisting of a timestamp plus a hash of the list of all revoked certificates, called certificate revocation list (CRL), and periodically send the signed CRL to the directories. A directory then just forwards that entire signed CRL to any user who requests the revocation status of a certificate. This approach is secure, but it is inefficient: the update and query authentication information has size $\Theta(n)$. Moreover, to turn the CRL approach into a persistent authenticated dictionary requires that every CRL ever issued be archived at the directories. Thus, if m CRLs have been issued, this solution requires in the worst case quadratic $O(m^2)$ storage space at each directory. In other words, the CRL-based approach is a simple but very inefficient solution for the persistent authenticated dictionary problem.

There are other more space-efficient methods for implementing ephemeral authenticated dictionaries. But methods for adapting them to the persistent context are not obvious. The *hash tree scheme* introduced by Merkle [13] can be used to implement a static authenticated dictionary, which supports the initial construction of the data structure followed by query operations, but not update operations (without complete rebuilding). Still, the only obvious way to make a hash tree persistent is to checkpoint the entire tree at each time quantum, which is clearly not space efficient.

Kocher [12] also advocates a static hash tree approach for realizing an authenticated dictionary, but simplifies somewhat the processing done by the user to validate that an item is not in the set S . Using techniques from incremental cryptography, Naor and Nissim [15] dynamize hash trees to support the insertion and deletion of elements using 2-3 trees. Other certificate revocation schemes based on variations of hash trees have been recently proposed in [2,5,10], as well, but do not deviate significantly from the above approaches.

Goodrich and Tamassia [6,7] have proposed an authenticated-dictionary scheme based on the skip-list data structure that has asymptotically the same performance as [15] but it is simpler to implement. Still, like other previous solutions, their data structure is ephemeral—it only stores the most recent copy of the set S .

Persistent Data Structures. Researchers have worked on persistent data structures for other abstract data types besides authenticated dictionaries. The idea of path copying in a tree, for example, which is a component in some of our solutions, has been used in non-authenticated contexts by several researchers (e.g. Myers [14] and Reps, Teitelbaum and Demers [17]). Sarnak and Tarjan [18] proposed the node-copying method and use persistent trees to solve the planar point location problem, while Driscoll, Sarnak, Sleator and Tarjan [4] developed techniques for making linked data structures persistent. Nevertheless, none of these previous schemes for making data structures persistent have directly addressed the need for authentication or directory distribution.

1.3 Summary of Results

We present two data structures for implementing PADs, based on the dictionary data structures known as red-black trees and skip lists. Our solutions allow for element insertions and removals in the current set S to run in $O(\log n)$ time and queries in the past to run in $O(\log m)$ time. More importantly, our solutions use only $O(\log n)$ additional space per update. (Recall that n is the number of elements in the current set S and m is number of updates that have occurred so far.) Thus, our solutions are significantly more efficient than the CRL-based approach or checkpointing-based approaches, which require $\Theta(n)$ additional space for archiving S at each historic time quantum. We describe the theoretical foundations behind our solution to the persistent authenticated dictionary problem in Section 2.

In addition, we claim that our methods are simple, which is an often neglected, but important aspect of computer security solutions, for implementation correctness is as important as theoretic soundness. To support this claim, we have implemented our solutions and have performed a number of benchmarking tests, which we report on in Section 3.

2 Making Authenticated Dictionaries Persistent

We use and extend some ideas from previous work on persistent data structures to create our solutions to the persistent authenticated dictionary (PAD) abstract data type. Let us therefore begin with a quick review.

A Quick Review of Persistent Data Structures. Most of the data structures are *ephemeral* in the sense that whenever the user performs an update to them he destroys the previous version. If the previous version is retained and we can do queries to them we talk about *persistent* data structures. We can even talk about fully persistent data structures if we can make updates and not only queries to previous versions.

In more detail, the operations that a persistent data structure S supports are **find**(e, t), which determines whether element e was in S at time t , **insert**(e), which inserts element e into S (at current time), and **delete**(e), which removes element e from S (at current time).

In our case, we maintain the data structure in both the source and the directories. When the user queries a directory if an element e was in the source at some time t , the directory returns “yes” if e existed in the source at time t along with a proof of its existence, or “no” if e did not exist in the source at time t along with a proof of its nonexistence. The proof must permit the user to verify that the answer is authentic (i.e., it is as trustworthy as if it were directly signed by the source) and that it is current (i.e., it corresponds to time t).

Depending on the application we can use one of the two versions of the PADs which differ in the way they define time. In particular we can have:

- *Discrete time* where we can think that the dictionary holds several versions whose time is numbered sequentially with integers starting at 0. When a user

performs a query, t equals the time of one of the versions, and the dictionary searches the corresponding version and returns an answer that allows the user to verify that indeed the dictionary searched version at time t .

- *Continuous time* where we can define the time over any set for which there exist a complete order. The versions of the dictionary are ordered according to that order. Whenever a user queries the directory about time t , the directory must provide an answer corresponding to the latest dictionary version at time t' that is earlier than t ; moreover, it must provide information for the user to verify that indeed t' is the time of the version that should have been queried and that does not exist a version at time t'' such that $t' < t'' \leq t$.

We have developed two data structures that implement PADs, one based on the red-black tree and the other based on the skip-list data structures.

2.1 PADs Based on Red-Black Trees

We denote the element stored at a node v as $\text{elem}(v)$.

The version of the red-black tree we use, has all the values stored only at the external nodes. We use the internal nodes to make queries to the tree and to store authentication information. The value stored at each internal node equals the maximum value of the left subtree; the values at the right subtree are greater than that value. When we want to make an insertion to the tree of an element with value e that does not exist into the tree, first we find the external node v containing the minimum element $\text{elem}(v)$, such that $\text{elem}(v) > e$. We then create a new internal node w to replace v and we set as the left child of w a new external node u such that $\text{elem}(u) = e$ and as the right child the node v .¹ Finally we perform the necessary reconstructions and recolorings. A deletion of a node is performed in the opposite manner.

The persistent red-black tree is a modification of the red-black tree. Each node u has also a field $\text{time}(u)$ where we store the timestamp of its creation. Each addition or deletion to the tree does not change the current tree nodes but instead it adds new nodes with timestamp value that of the current time.

In Fig. 2 we can see an example of an instance of a red-black tree. The bold lines denote black nodes and edges, while the normal ones denote red. The label at the top of each node (in Fig. 2 all are 0 for simplicity) denotes the timestamp value.

Suppose we want to add a new element ($e = 18$). We can see the new tree created in Fig. 3, where the timestamp of the new nodes has taken the current time value (here just 1 for simplicity). Because at each update operation we copy the whole path from a node up to the root, this method is called *path-copying* method. Note that we have as many signed roots as update operations. Details of the operations follow in the next paragraphs.

In our scheme we use an authenticated persistent red-black tree. Each node u has stored one more value, $\text{auth}(u)$, that is used for authentication; $\text{auth}(u)$

¹ There is an exception for the value that is larger than the largest element at the tree.

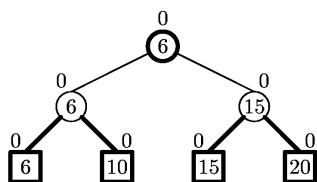


Fig. 2. The tree before the insertion.

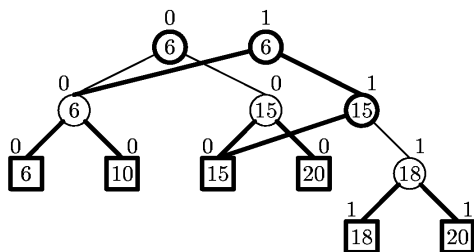


Fig. 3. The tree after the insertion of element 18.

equals $\text{elem}(u)$ if u is an external node, and the concatenation of the $\text{auth}()$ values of u 's children if u is an internal node. h must be a collision-free hash function. Finally the root of every version of the tree (for the different update times) is signed. In summary, each node u contains three fields: $\text{elem}(u)$, $\text{time}(u)$, and $\text{auth}(u)$.

We describe now the details of the operations.

Insertion. At every **insert** operation, the old version of the tree doesn't change at all (except for the colors of the nodes and edges as we will see later). We find the external node u that will be the father of the new node. Then we make a copy of the nodes on the path, from the root down to u to which we assign a timestamp value equal to the current time. We perform then the insertion by creating a new node as we saw before. We denote here a node x as x_0 for the old version, and x_1 for the new version that is copied. If along the path, a node v that is copied has a child w that is also copied, we add the edge from the node v_1 to w_1 . For the rest of the edges of v_0 to some other node z_0 we add an edge from v_1 to z_0 . We can see an example in Figures 2 3.

Since we have a red-black tree, we must color the new nodes. Every node v_1 will be colored with the same color with v_0 , and the insertion will be done normally. After the insertion we may have to perform some transformations on the tree, namely some rotations and recolorings. The recolorings are easy, since we can overwrite the old colors of any node of any previous version of the tree. The colors are used only for the update operations, so we do not need to keep track of the old ones—we do not make updates in the past. Rotations are also simple; they will be done only on the new version of the tree and they will affect only the new nodes and the links originating from them.

Finally we must compute the values $\text{elem}()$ and $\text{auth}()$ for all the new nodes created. These operations are straightforward and we can perform them as we create the new nodes, in $O(\log n)$ time. In particular, we must compute only two $\text{elem}()$ values, the one of the new external node, and that of its parent (which has the same value). We must compute the $\text{auth}()$ values of all the $O(\log n)$ new nodes. Finally, if r_1 is the root of the last version of the tree (just created) and r_0 is the root of the just previous version, then the source

signs $(\text{auth}(r_1), \text{time}(r_1), \text{time}(r_0))$. (The previous timestamp is necessary only in the case of continuous time; we give more details later.)

Deletion. A **delete** operation is similar to an **insert**. We copy the path from the root down to the node to be deleted, we remove the nodes (one external and one internal node will be deleted), and then perform the necessary rotations and recolorings. Here we must be careful, since the rotations in this case may also affect nodes not existing in the new path created. So before the rotation, these nodes must also be copied.

During the deletion we may require to perform in the worst case at most a logarithmic number of recolorings and two rotations. In total, except for the path of the node deleted, it may be necessary to copy at most 4 additional nodes; therefore, we copy only a logarithmic number of nodes.

Finally we compute the necessary $\text{elem}()$ and $\text{auth}()$ values and sign the new root, by the method we described in the *Insertion* paragraph. Again, we must compute all the $\text{auth}()$ values of the $O(\log n)$ new nodes, and the only internal $\text{elem}()$ value that we must alter is the value of the node that followed the removed external node in the infix traversal of the tree.

Query. A **find** operation can query the PAD at any time in the past (or present) to find if an element e existed at that time. Say that we want to query for time t_1 and we have the two consecutive tree versions for times t_0 and t_2 , where $t_0 \leq t_1 < t_2$, assuming that t_2 exists. In the following, assume that r_0 is the root with timestamp t_0 , r_2 is the root with timestamp t_2 and r_3 is the root with the largest timestamp that is less than t_0 , if it exists. In order to search for e we work on the tree originating from the root r_0 of time t_0 . We distinguish two cases:

- **The element e exists in the tree at time t_0 :** The dictionary returns to the user the answer “yes”, the root r_0 signed by the source (i.e., the signature of $(\text{auth}(r_0), \text{time}(r_0), \text{time}(r_3)))$, the root r_2 signed by the source, that is, the signature of $(\text{auth}(r_2), \text{time}(r_2), \text{time}(r_0))$, the element e , and the sequence $Q(e)$ of the hashes of the siblings of the nodes of the path from the root of the tree to e . The user can verify that the answer is valid from the sequence $Q(e)$ and the source’s signature of the root.
- **The element e does not exist in the tree at time t_0 :** In this case, let e' be the maximum element that exists in the tree and is smaller than e , and e'' the minimum element that exists in the tree and is greater than e . The dictionary returns to the user the answer “no”, the root r_0 with timestamp t_0 and the root r_1 with timestamp t_1 , signed by the source like in the previous case, the elements e' and e'' , and the sequences $Q(e')$ and $Q(e'')$ defined as before. The user can verify now that both e' and e'' exist in the tree and that they are consecutive external nodes (hence e is not between them).

The root r_2 has to be included only in the case of continuous time. The reason is to prove that there is not another version of the tree at time t_4 such that $t_0 < t_4 \leq t_1 < t_2$ which may have different information concerning C .

The user can now verify that the version that immediately follows the one that corresponds to time t_0 is at time t_2 since it is included in the signature.

Summarizing the above results, we have the following theorem:

Theorem 1. *A persistent authenticated dictionary can be implemented with a persistent authenticated red-black tree. The time needed for an update operation is $O(\log n)$ and for a query operation is $O(\log m)$, where n is the number of elements of the last version and m is the total number of versions. The total space requirement is $O(\sum_{i=1}^m \log n_i)$, where n_i is the number of elements of version i .*

Proof. We perform an update to the last version of the tree, which has n elements. The update operation requires the addition of $O(\log n)$ new nodes, $O(\log n)$ changes to the tree structure (including recolorings) and the computation of $O(\log n)$ new hash values. Therefore, it can be performed in time $O(\log n)$.

A query on version i requires a binary search on the root of the trees to find the appropriate version of the tree, which takes time $O(\log m)$, a search in the tree to find the appropriate node(s), which needs $O(\log n_i)$ steps, and the creation and return of the response, which is done in $O(\log n_i)$ steps. Since n_i cannot be greater than m , the overall time requirement of a query is $O(\log m)$.

The space required for the i -th update is $O(\log n_i)$: the insert operations need to copy only the nodes of the path of the new node, while the delete operations may need to copy at most 4 additional nodes. By the property of red-black trees the total path length is $O(\log n_i)$, hence, the total space is $O(\sum_{i=1}^m \log n_i)$. \square

Note that the above time complexity results hold when we have continuous time. In the case of discrete time, we can find the appropriate root for a query just by a table lookup in constant time, and in this case the total query time is $O(\log n_i)$. Also, note that when we query the last version, the time is reduced to $O(\log n)$, as in the ephemeral authenticated dictionary.

2.2 PADs Based on Skip Lists

The *skip-list* data structure [16] is an efficient means for storing a set S of elements from an ordered universe. Briefly, a skip list consists of a collection of linked lists S_0, S_1, \dots, S_k (where S_i contains a randomly selected subset of the items in S_{i-1} , plus two additional values $-\infty$ and $+\infty$), and links between them. We can see an example in Fig. 4. With high probability (whp [8]), skip lists have the same asymptotic performance as red-black trees. However, experimental studies (e.g., see [16]) have shown that they often outperform in practice 2-3 trees, red-black trees, and other deterministic search tree structures.

Commutative Hashing. To simplify the verification process, *commutative cryptographic hash functions* are introduced in [6]. A hash function h is commutative if $h(x, y) = h(y, x)$ for all x and y . A commutative hash function is *commutatively collision resistant* [6] if, given (a, b) , it is difficult to compute a pair

² We use “whp” to refer to a probability that is at least $1 - 1/n^c$ for some constant $c \geq 1$.

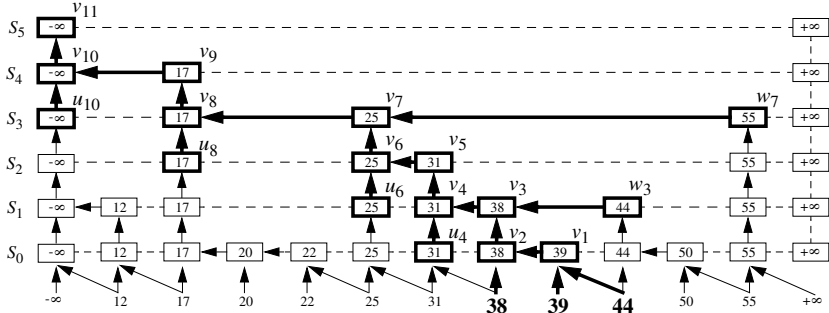


Fig. 4. (a) An example of a skip list. All the lines indicate links of the data structure. (b) The answer authentication information for the presence of element $x = 39$ (and for the absence of element 42) consists of the signed time-stamped value $f(v_{11})$ of the source element and the sequence $Q(x) = (44, 39, 38, f(w_3), f(u_4), f(u_6), f(w_7), f(u_8), f(u_{10}))$. The user recomputes $f(v_{11})$ by accumulating the elements of the sequence with the hash function h , and verifies that the computed value of $f(v_{11})$ is equal to the value signed by the source [6,7]. The arrows denote the flow of authentication information.

(c, d) such that $h(a, b) = h(c, d)$ while $(a, b) \neq (c, d)$ and $(a, b) \neq (d, c)$. Given a cryptographic hash function g that is collision resistant in the usual sense, the commutative hash function, $h(x, y) = g(\min\{x, y\}, \max\{x, y\})$ is commutatively collision resistant if x and y have the same length [6].

Authenticated Dictionary Based on a Skip List. Using the skip-list data structure and commutative hashing, we can design a scheme for authenticated dictionaries, similar to the one based on balanced trees. It has the same asymptotic performance but it avoids many complications that arise in the implementation of the hash trees, leading to easier and less error-prone implementations. An example of the data structure and of an authenticated query is in Fig. 4. For more details refer to [6,7].

PAD Based on a Skip List. We can apply the path-copying idea to the authenticated skip list and create a persistent authenticated skip list. First, we make the following observation, which allows us to have an efficient implementation: *for the operations supported by the skip list, some of the nodes and links are unnecessary and can be omitted.* The skip list then looks like that of Fig. 5. We can see now that in this form the skip list is essentially a binary tree (whose root is the node in the highest-level list with the value $-\infty$) and so we can apply the path-copying method. We have to be careful however to copy all the nodes whose out-links or authentication information change.

It is important to mention that when we do an update, we do not change the existing nodes of the skip list—we only add new nodes. Therefore, by storing links to the appropriate root nodes we can make queries to any version of the dictionary. We describe now in detail the operations.

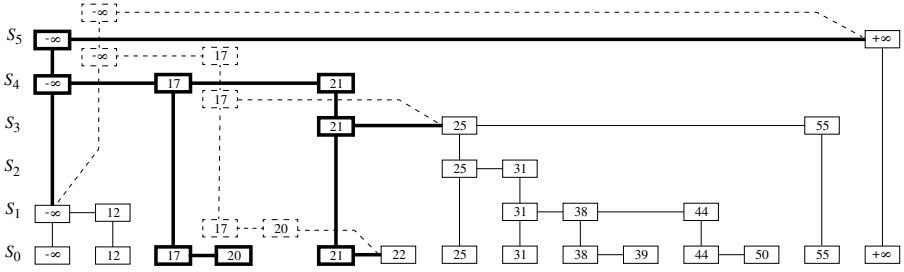


Fig. 5. (a) A tree-like representation of a skip list. (b) Insertion of element 21 with height 5. The new nodes and links are bold, while the dashed ones exist only in the previous version.

Insertion. Assume that we want to insert element e and that e' is the largest element that is smaller than e . On the search path p we will reach element e' that belongs to the list S_0 . We insert the element e and hence the authentication data of e' in S_0 changes; the change of the authentication data propagates up to the root node *along the path p that we followed to reach e'* . In other words, the nodes of p (and of course the nodes of the new element) are the only nodes whose authentication data changes. Finally, some of the links to the right of the nodes on p —basically, those that belong to nodes that are lower than the height of the new element—must be set to null and be replaced by links that originate from the new elements. Therefore, the new nodes that we create are exactly those that belong to p plus those of the new element e . Fig. 5 shows an example of an insertion.

The number of the new nodes that we create equals the length of the search path to the element e' plus the number of the new nodes for the new element. Both of these are $O(\log n)$ whp. The number of hashes that we have to compute is at most equal to the number of the new nodes (although for some of the nodes the authentication data is equal to that of the node below and does not have to be recomputed).

Deletion. Assume that we want to remove element e from the skip list and that e' is the element immediately preceding e . We follow the path p to element e' . Since we remove the element next to e' (i.e., e), the authentication data of e' in S_0 changes. Again, the change propagates up to the root node only along p , and the only links to the right that must change are in p . Therefore, the only nodes that we must duplicate are those in p , which with high probability are $O(\log n)$. Also, the number of the hashes to compute equals at most the number of the new nodes and so it is $O(\log n)$ whp.

Query. Assume that we want to find if element e existed in the skip list at time t . First, we find the appropriate root node r of the skip list. Like in the red-black trees implementation, in the case of discrete time, we can find the start node by a table lookup in constant time, while for continuous time we can find it in time

$O(\log m)$ using, for example, binary search. Since in the update operations we never change existing nodes but we only add new ones, the skip list originating from r is exactly the one we had at time t . Therefore, we can perform the query simply by following the procedure of the ephemeral authenticated skip list; the time needed is $O(\log n_t)$, where n_t is the number of elements of the skip list of version at time t .

Thus, the total time needed is $O(\log n_t)$ for discrete time and $O(\log m)$ for continuous-time.

Theorem 2. *A persistent authenticated dictionary can be implemented with a persistent authenticated skip list. With high probability, the time needed for an update operation is $O(\log n)$ and for a query operation $O(\log m)$, where n is the number of elements of the last version and m is the total number of versions. After an update, the space used by the data structure increases by $O(\log n)$ whp.*

2.3 Security

The security of both techniques for PADs is based on Merkle's scheme for digital signatures [13] which is the cryptographic basis for previous approaches for authenticated dictionaries as well [12,15,2,5,10,6].

One possible way for the directory to cheat is to try to return to the user a response corresponding to a different version than the specified. In the discrete-time setting, the response of the directory must contain a signed statement of the source that contains both the number of the version that the user specified at his query and the hash value of the corresponding root. In the continuous-time setting, as we explained in section 2.1, the response of the directory allows the user to verify that indeed the hash of the correct root is returned. Hence, in both cases, the user is assured (based only on the trust in the source) that the hash value of the root that he received is that of the correct version.

Assuming that the directory returns the authentication information corresponding to the correct version, our schemes for PADs are as secure as the corresponding ephemeral ones: In order for the directory to cheat, it has to find a sequence of $O(\log n)$ values, which, if are hashed in succession, produce the same output as the values that actually exist in the dictionary; if we use a collision-resistant cryptographic hash function, such as MD5 or SHA1, this task is infeasible. For more details, refer to [13,15,6].

2.4 Extensions

We present here three useful extensions of the preceding schemes that can be implemented with minor overhead.

First, instead of performing membership queries, we can have a value associated with each element to be returned by a query operation. The element defines the position of the pair in the data structure, and the authentication data depends on both the element and the value. The additional operation that we allow is changing the value that corresponds to an element and we can implement it easily by only copying the old nodes (no restructural operations are necessary).

Second, we can apply the path-copying technique to other types of authenticated balanced trees, such as AVL or 2-3 trees.

Finally, we can combine the advantages of the PADs with the efficiency offered by *B*-trees when we store them in secondary memory. This yields an efficient authenticated data structure that is persistent in two ways: it holds the whole history of updates, and remains in permanent, external memory—usually a disk. The idea is to allocate a whole new disk block when we create a new node, and in some time instances to compact older versions, by putting two small nodes in the same block, to save disk space.

3 Experimental Results

We have designed a comprehensive set of Java interfaces describing the PAD abstract data type and auxiliary data types (source, directory, user, etc.). Also, we have conducted an experiment on the performance of our data structures on randomly generated sets of 256-bit integers in dictionaries ranging in size from 0 to 500,000. For each operation, the average was computed over 10,000 trials. The experiment was conducted on a 400MHz Sun Ultra Enterprise with 2GB of memory running Solaris. The Java Virtual Machine was launched with a 1GB maximum heap size. Cryptographic hashing was performed using the standard Java implementation of the MD5 algorithm. The signing of the update authentication information by the source and the signature verification by the user were omitted from the experiment.

We compare the execution time of the ephemeral authenticated skip list with that of the persistent one. The highest level of a tower was limited to 20. The results are shown in Fig. 6. We can see that the query time is almost the same for both the ephemeral and persistent versions, while the insertion time is slightly lower for the ephemeral one, since it allows some optimizations in the implementation.

After noticing that most of the update time (about 94%) is consumed by the hashing computations, we have determined the number of hashes required for an update in a skip list (it is the same for both the ephemeral and the persistent versions) and in a red-black tree. We present the experimental results in Fig. 6. Furthermore, we observe that the number of hashes computed at each update equals the number of new nodes created in a red-black tree, and is about $0.25 \log_2 n$ less than the number of new nodes created in a skip list. Therefore, the number of hashes provides a measure of the space requirement of our data structures. From Fig. 6, we can see that the red-black tree requires fewer hashes than the skip list. This is justified theoretically, since for a skip list with n elements, the average number of comparisons performed in a search (which is a lower bound of the number of hashes that we have to compute for an update) is $\frac{3}{2} \log_2 n + \frac{7}{2}$ [16], while for a balanced tree with n external nodes, it is about $\log_2 n$ [11].

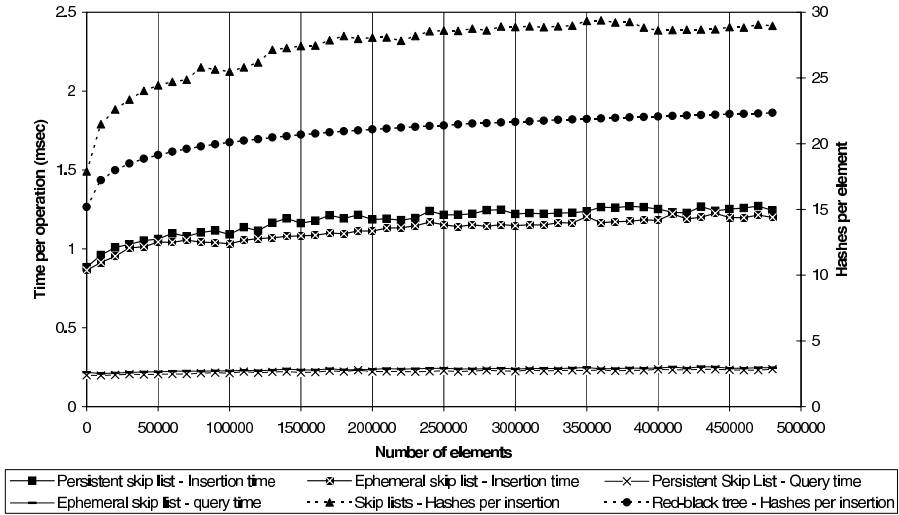


Fig. 6. (a) Average insertion and query times for ephemeral and persistent authenticated skip lists. The results for deletions are similar to those for insertions. (b) Average number of hash computations per insertions for skip lists (both persistent and ephemeral) and red-black trees.

4 Conclusions

We have introduced the notion of persistent authenticated dictionaries and justified their usefulness. We have also presented two techniques for implementing persistent authenticated dictionaries that support updates in $O(\log n)$ time and queries in $O(\log m)$ time, and use $O(\log n)$ space per update (n is the number of elements in the current set S and m is the number of updates that have occurred so far). The technique based on red-black trees has better running time and space requirement than the one based on skip lists, but its implementation is more complex. We leave as an open problem whether persistent authenticated dictionaries can be implemented with constant space per update.

Acknowledgments. We would like to thank Andrew Schwerin for many useful conversations regarding the contents of this paper.

References

- [1] W. Aiello, S. Lodha, and R. Ostrovsky. Fast digital identity revocation. In *Advances in Cryptology – CRYPTO ’ 98*, Lecture Notes in Computer Science. Springer-Verlag, 1998.
- [2] A. Buldas, P. Laud, and H. Lipmaa. Accountable certificate management with undeniable attestations. In *ACM Conference on Computer and Communications Security*. ACM Press, 2000.
- [3] P. Devanbu, M. Gertz, C. Martel, and S. Stubblebine. Authentic third-party data publication. In *Fourteenth IFIP 11.3 Conference on Database Security*, 2000.

- [4] J. R. Driscoll, N. Sarnak, D. D. Sleator, and R. E. Tarjan. Making data structures persistent. *J. Comput. Syst. Sci.*, 38:86–124, 1989.
- [5] I. Gassko, P. S. Gemmell, and P. MacKenzie. Efficient and fresh certification. In *International Workshop on Practice and Theory in Public Key Cryptography '2000 (PKC '2000)*, Lecture Notes in Computer Science, pages 342–353, Melbourne, Australia, 2000. Springer-Verlag, Berlin Germany.
- [6] M. T. Goodrich and R. Tamassia. Efficient authenticated dictionaries with skip lists and commutative hashing. Technical Report, Johns Hopkins Information Security Institute, 2000.
- [7] M. T. Goodrich, R. Tamassia, and A. Schwerin. Implementation of an authenticated dictionary with skip lists and commutative hashing. In *Proc. 2001 DARPA Information Survivability Conference and Exposition*, volume 2, pages 68–82, 2001.
- [8] C. Gunter and T. Jim. Generalized certificate revocation. In *Proc. 27th ACM Symp. on Principles of Programming Languages*, pages 316–329, 2000.
- [9] C. Kaufman, R. Perlman, and M. Speciner. *Network Security: Private Communication in a Public World*. Prentice-Hall, Englewood Cliffs, NJ, 1995.
- [10] H. Kikuchi, K. Abe, and S. Nakanishi. Performance evaluation of certificate revocation using k -valued hash tree. In *Proc. ISW'99*, volume 1729 of *LNCIS*, pages 103–117. Springer-Verlag, 1999.
- [11] D. E. Knuth. *Sorting and Searching*, volume 3 of *The Art of Computer Programming*. Addison-Wesley, Reading, MA, 1973.
- [12] P. C. Kocher. On certificate revocation and validation. In *Proc. International Conference on Financial Cryptography*, volume 1465 of *Lecture Notes in Computer Science*, 1998.
- [13] R. C. Merkle. A certified digital signature. In G. Brassard, editor, *Advances in Cryptology—CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*, pages 218–238. Springer-Verlag, 1990.
- [14] E. W. Myers. Efficient applicative data types. In K. Kennedy, editor, *Conference Record of the 11th Annual ACM Symposium on Principles of Programming Languages*, pages 66–75, Salt Lake City, UT, Jan. 1984. ACM Press.
- [15] M. Naor and K. Nissim. Certificate revocation and certificate update. In *Proceedings of the 7th USENIX Security Symposium (SECURITY-98)*, pages 217–228, Berkeley, 1998.
- [16] W. Pugh. Skip lists: a probabilistic alternative to balanced trees. *Commun. ACM*, 33(6):668–676, 1990.
- [17] T. Reps, T. Teitelbaum, and A. Demers. Incremental context-dependent analysis for language-based editors. *ACM Transactions on Programming Languages and Systems*, 5(3):449–477, July 1983.
- [18] N. Sarnak and R. E. Tarjan. Planar point location using persistent search trees. *Commun. ACM*, 29(7):669–679, July 1986.

Efficient Optimistic N-Party Contract Signing Protocol

Josep L. Ferrer-Gomila, Magdalena Payeras-Capellà, and Llorenç Huguet-Rotger

Departament de Ciències Matemàtiques i Informàtica, Universitat de les Illes Balears,
Carretera de Valldemossa km. 7.5, Palma de Mallorca, 07071, Spain
dijjfg@clust.uib.es

Abstract. Contract signing is a fundamental service in the environment of the electronic commerce. It is a kind of fair exchange of values: a signature on a text contract for another signature on the same text. We present a fair protocol that requires the existence and possible involvement of a TTP (trusted third party), but it only intervenes in case of exception (it plays a subsidiary role). The protocol is the best solution to the date in terms of efficiency. We present and analyse a two-party and three-party version of the protocol, and we outline an N-party version. Finally, it is very simple and easy to understand. If we think that at the end some conflicts will be solved in courts, it is a non worthless characteristic.

1 Introduction

The World Wide Web, e-mail, EDI, etc, have proven to be very effective in e-commerce environments. And contract signing becomes a fundamental electronic service. It is clear that it will save time, money, etc., but we have to face with a problem: lack of security. Electronic contract signing is one service offered to users, such as they want to obtain a signed copy of a contract from another party.

Protocols for contract signing have to provide irrefutable evidence to parties to prove, at the end of the exchange, if the contract is signed and the terms of the contract. We handle contract signing as a fair exchange of values: the originator has an item (a text of a contract and a non-repudiation token) to be exchanged for a recipient's item (a non-repudiation token bounded to the text of the contract). An exchange is fair if at the end, either each player receives the item it expects or neither player receives any additional information about the other's item [2].

One kind of solutions is based on the gradual release of secrets (parties exchange non-repudiation tokens "simultaneously"). This approach [9, 15] achieves fairness by the gradual release of information over many rounds: some knowledge about the non-repudiation evidence is revealed during each round. It seems to be too cumbersome and inefficient for actual implementation, due the high communication overhead. Moreover, fairness is based on the assumption of equal computational power. This assumption is unrealistic in practice [3].

A second kind of solutions are the third party protocols (parties exchange items, assisted by a TTP). We can find some fair exchange protocols that make use of a TTP [1, 2, 4, 5, 7, 8, 10, 11, 13, 17, 19, 20, 21]. They differ in the degree of TTP's involvement in a protocol run. Following this viewpoint (TTP's involvement), we

classify the protocols into two classes: with active TTPs (a TTP is actively involved in every protocol run) and with subsidiary TTPs or optimistic protocols [1] (a TTP only intervenes in the case of an exception occurs, and so not in every protocol run).

Some formal requirements for optimistic fair exchange were stated in [2], and reformulated in [21]: effectiveness, fairness, timeliness, non-repudiation and verifiability of third party. Two additional properties to be met are efficiency and privacy (this last one is optionally for users). [22] adds a last property: abuse-free. They define this property as follows: "An optimistic contract signing protocol is abuse-free if it is impossible for a single player at any point in the protocol to be able to prove to an outside party that he has power to terminate (abort) or successfully complete the contract".

ASW [2] and GJM [22] protocols have, both, three sub-protocols: *exchange*, *abort* and *resolve*. In the normal case, only the *exchange* sub-protocol is executed, with four steps in both protocols. We agree with [21] that ASW protocol has some problems (the *abort* sub-protocol is flawed, etc.). As regards the GJM protocol, we think that they present a relatively complex solution to achieve the abuse-free property.

Finally, we think that solutions have to be secure from a technical point of view, but as simple as possible. Some conflicts will have to be solved in courts, and it means that a judge will have to evaluate evidence brought by parties involved in an electronic contract signing. Obviously, judges will have to be assisted by experts, but understanding, at the end, conclusions given by these experts.

In this paper we present a protocol for two-party contract signing with the following characteristics: effective, fair, asynchronous and efficient. Additionally we introduce a multi-party version of the previous contract signing protocol. This protocol is better than previous solutions at least in two aspects. On one hand, the steps of the basic protocol are less than any solution in the literature. On the other hand, as the two-party solution, it is easy to understand and so, it can be useful to be used in situations that potentially will arrive to courts.

2 An Efficient Protocol

We will begin describing the proposed two-party protocol. Then we will discuss the possible disputes resolution.

2.1 Protocol

The originator, *A*(lice), and the recipient, *B*(ob), will exchange non-repudiation evidence directly. Only in the case they cannot get the expected items from the other party, the TTP (*T*) will be invoked, by initiating *cancel* or *finish* sub-protocols. The notation and elements used in the protocol description are as follows:

- X, Y concatenation of two messages X and Y
- $H(X)$ a collision-resistant one-way hash function of message X
- $PR_i[H(X)]$ digital signature on message X with the private key, or signing key, of principal i (using some hash function, $H()$, to create a digest of X)

- $i \rightarrow j: X$ principal i sends message (or token) X to principal j
- M message containing the contract to be signed; the contract specifies who is the originator, $A(lice)$, and who is the recipient, $B(ob)$
- $h_A = PR_A[H(M)]$ signature of A on the contract M
- $h_B = PR_B[H(M)]$ signature of B on the contract M
- $ACK_A = PR_A[H(h_B)]$ signature of A on h_B ; this is an acknowledgement that A knows that the contract is signed, and is part of the necessary evidence for B
- $ACK_T = PR_T[H(h_B)]$ signature of TTP on h_B ; this is an equivalent acknowledgement to which A should have sent
- $h_{AT} = PR_A\{H[H(M), h_A]\}$ this token is an evidence that A has demanded TTP 's intervention
- $h_{BT} = PR_B\{H[H(M), h_A, h_B]\}$ this token is an evidence that B has demanded TTP 's intervention
- $h_B' = PR_T[H(h_B)]$ signature of TTP on h_B to prove its intervention

The *exchange* sub-protocol is as follows:

1. $A \rightarrow B$: M, h_A
2. $B \rightarrow A$: h_B
3. $A \rightarrow B$: ACK_A

If the protocol run is completed, the originator A will hold non-repudiation (NR) evidence, h_B , and the recipient B will hold non-repudiation evidence, h_A and ACK_A . So the protocol meets the effectiveness requirement. If it is not the case, A or B , or both, need to rectify the unfair situation by initiating the *cancel* or *finish* sub-protocol, respectively, so that the situation returns to a fair position.

If A “says” (A could be trying to cheat or being in a wrong conception of the exchange state) that she has not received message 2 from B , A may initiate the following *cancel* sub-protocol:

- | | |
|----------------------|------------------------------------------------------------|
| | 1'. $A \rightarrow T$: $H(M), h_A, h_{AT}$ |
| IF (finished = true) | 2'. T : retrieves h_B |
| | $T \rightarrow A$: h_B, h_B' |
| ELSE | 2'. $T \rightarrow A$: $PR_T[H(\text{“cancelled”}, h_A)]$ |
| | T : stores <i>cancelled</i> = true |

The TTP will verify the correctness of the information given by A . If it is not the case the TTP will send an error message to A . Otherwise, it will proceed in one of two possible ways. If the variable *finished* is *true*, it means that B had previously contacted with the TTP (see paragraph below). The TTP had given the NR token to B , ACK_T , and now it has to give the NR token to A . So, it retrieves this stored NR token, h_B , and sends it to A , and a token to prove its intervention, h_B' . If B had not contacted with the TTP previously, the TTP will send a message to A to *cancel* the transaction, and it will store this information (*cancelled* = *true*) in order to satisfy future petitions from B . Whatever case, now, we are again in a fair situation.

If B “says” that he has not received message 3, B may initiate the following *finish* sub-protocol:

	2'. $B \rightarrow T: H(M), h_A, h_B, h_{BT}$
IF (cancelled = true)	3'. $T \rightarrow B: PR_T[H(\text{“cancelled”}, h_B)]$
ELSE	3'. $T \rightarrow B: ACK_T$
	$T: \text{stores } finished = true, \text{ and } h_B$

The TTP, also, will verify the correctness of the information given by B . If it is not the case the TTP will send an error message to B . Otherwise, it will proceed in one of two possible ways. If the variable *cancelled* is *true*, it means that A had previously contacted with the TTP (see paragraph above). The TTP had given a message to A to *cancel* the transaction, and now it has to send a similar message to B . If A had not contacted with the TTP previously, the TTP will send the NR token, ACK_T , to B . In this case the TTP will store the NR token, h_B , and will assign the value *true* to the *finished* variable, in order to satisfy future petitions from A . Again, whatever case, now, we are in a fair situation.

Observe that a “conflicting” situation may occur. A can obtain NR evidence from B (h_B) and a *cancel* message from T , while B obtain NR evidence from A (h_A, ACK_A). She can do it, for instance, invoking the *cancel* sub-protocol after the end of the *exchange* sub-protocol. It seems that A can affirm that the contract is signed or is not signed, depending on her usefulness. But B possesses NR evidence that will prove that the contract is signed, and if A tries to use the *cancel* message she will be proving she is a cheating party. As a conclusion, the protocol is fair and we have not made timing assumptions (the protocol is asynchronous, parties can contact with TTP when they want). You can find a discussion about verifiability of TTP in Appendix 1.

2.2 Dispute Resolution

After a protocol run is completed (with or without the participation of the TTP), disputes can arise between participants. We can face with two possible types of disputes:

- repudiation of A : B claims that the contract is signed; and
- repudiation of B : A claims that the contract is signed.

An external arbiter (not part of the protocol) has to evaluate the evidence held and brought by the parties, to resolve these two types of disputes. As a result of this evaluation, the arbiter will determine who has right on its side. The arbiter has to know who is the originator and who is the recipient; remember that the contract, M , contains this information.

In the case of repudiation of A , B is claiming that he received the signature on the contract M from A . He has to provide the following information to an arbiter: M , h_A and ACK_A or ACK_T . The arbiter will check if h_A is A 's signature on M , and if it is positive the arbiter will assume that A had sent her signature to B . Then, the arbiter will check if ACK_A is A 's signature on h_B , or it will check if ACK_T is TTP's signature on h_B . If this check is positive, the arbiter will assume that either A or the TTP had sent an acknowledgement to B . Therefore, the arbiter will side with B . Otherwise, if one or both of the previous checks fails, the arbiter will reject B 's demand. If the

evidence held by B proves he is right, and A holds a message like $PR_T[H(\text{"cancelled"}, h_A)]$, it means that the TTP or A had acted improperly (see Appendix 1).

In the case of repudiation of B , A is claiming that B had signed the contract M . She has to provide the following information to an arbiter: M and h_B . The arbiter will check if h_B is B 's signature on M , and if it is positive the arbiter will assume that B had received M and h_A , and that he is committed to obtain the acknowledgement, ACK_A or ACK_T . If the previous check fails, the arbiter will reject A 's demand. If the check is positive, the arbiter should interrogate B . If B contributes a *cancel* message, it means that B contacted with the TTP, and the TTP observed that A had already executed the *cancel* sub-protocol. For this reason the TTP sent the *cancel* message to B . Now it is demonstrated that A has tried to cheat. Therefore, the arbiter will reject A 's demand, and the arbiter will side with B . If B cannot contribute the *cancel* message, the arbiter will side with A .

As a conclusion, the protocol meets the non-repudiation requirement.

2.3 A Three-Steps Asynchronous Protocol

Theorem 2 of [23] states: "There is no asynchronous optimistic contract signing scheme with three messages in the optimistic case". Now, we will show that it is not always true; at least not in all conditions.

A common criticism to the presented protocol is the one that is described next. Assume A is dishonest, sends the first message to B , *cancels* the contract with TTP, receives second message from B , and stops. If B contacts with TTP it will get a *cancel* message, which *seems* inconsistent with the fact that A already "has" the contract. But, look at the previous section (2.2) and you will see that B can prove that A is a cheating party, and so, B is not compelled to that contract. So, and very important, the three-steps asynchronous protocol is fair, even if A tries to cheat.

A second criticism is the following. Assume B needs to present a proof that A signed the contract to another party C . Somebody could think: " C needs not only to verify the signature of A , but C has also to contact TTP that A did not *cancel* the signature". It is not true: if B has first and third message (this last message from A or from TTP) the contract is signed (A can not *cancel* the contract).

A third (own) criticism is as follows. Assume A needs to present a proof that B signed the contract to another party C . C needs not only to verify the signature of B , but C has also to contact TTP (or B) that A did not *cancel* the signature. It is true. A can have the second message from B , but she can also have a *cancel* message obtained from the TTP without sending third message to B , and so the contract is not signed. As a conclusion C needs to verify with TTP or B if the contract is signed or not, but the protocol remains fair.

So the protocol is efficient (only three steps) and remains fair, optimistic and asynchronous.

3 Multi-party Version

A multi-party version is necessary when more than two parties have to be involved in the same contract signing process, and every party wants to be bound to the contract if

all parties are bound at the end of the exchange. It is to say, nobody wants a partially signed contract. A few solutions [6, 12, 13, 14, 22] have been presented to the date. These solutions are difficult to understand and not very efficient. Here we present an extension of the two-party protocol in section 2. A three-party version will be introduced, and a N-party will be pointed out.

3.1 A Three-Party Version

Parties, A (lice), B (ob) and C (harles), as in the two-party version, will exchange messages and non-repudiation evidence directly. Only as a last recourse, in the case they cannot get the expected items from the other parties, the TTP (T) will be invoked, by initiating *cancel* or *finish* sub-protocols. Messages are over-simplified to stand a clear explanation (aspects as confidentiality, messages linking in an exchange, verifiability of the TTP, etc, are out of the scope of this brief explanation). The elements used in the protocol description are as follows:

- M message containing the contract to be signed; the contract specifies who is the originator, A (lice), and who are other parties, B (ob) and C (harles), specifying the order in a ring
- $h_i = \text{PR}_i[H(M)]$ signature of principal i on the contract M
- $\text{ACK}_A = \text{PR}_A[H(h_B, h_C)]$ signature of A on h_B and h_C ; A knows that one round is completed
- $\text{ACK}_B = \text{PR}_B[H(\text{ACK}_A, h_C)]$ signature of B ; B knows that the second round has begun
- $\text{ACK}_C = \text{PR}_C[H(\text{ACK}_A, \text{ACK}_B)]$.signature of C ; C knows that the second round has finished
- $\text{ACK2}_B = \text{PR}_B[H(\text{ACK}_C)]$ signature of B ; C will know that the process has finished
- $\text{ACK}_T = \text{PR}_T[H(M)]$ signature of TTP on $H(M)$; this is an equivalent acknowledgement to which parties should have sent (it is a *NR* evidence)

The *exchange* sub-protocol is as follows:

1. $A \rightarrow B$: M, h_A
2. $B \rightarrow C$: M, h_A, h_B
3. $C \rightarrow A$: h_B, h_C
4. $A \rightarrow B$: h_C, ACK_A
5. $B \rightarrow C$: $\text{ACK}_A, \text{ACK}_B$
6. $C \rightarrow A$: $\text{ACK}_B, \text{ACK}_C$
7. $A \rightarrow B$: ACK_C
8. $B \rightarrow C$: ACK2_B

If the protocol run is completed, everybody will hold non-repudiation (NR) evidence:

- A will hold $h_B, h_C, \text{ACK}_B, \text{ACK}_C$
- B will hold $h_A, h_C, \text{ACK}_A, \text{ACK}_C$
- C will hold $h_A, h_B, \text{ACK}_A, \text{ACK}_B, \text{ACK2}_B$

As you can see 8 steps are enough if parties behave correctly, in front of 17 steps for the exchange sub-protocol in [22]. If it is not the case, *A* or *B* or *C*, or some of them, need to rectify the unfair situation by initiating *cancel* or *finish* sub-protocols, so that the situation returns to a fair position. In every sub-protocol, the TTP will be involved, and first of all, TTP's intervention will be to verify the correctness of the information given by parties. If it is not the case, it will send an error message to that party (we will no repeat this question in every case). We will need some state variables (*cancel*, *finish*, *cancel_A* and *cancel_C*), all of them with value *false* at the beginning for a particular exchange.

If *A* "says" that she has not received message 3 from *C*, *A* may initiate the following *cancel* sub-protocol:

	1'. $A \rightarrow T: H(M), h_A$
IF (<i>finish</i> = true)	2'. $T \rightarrow A: ACK_T$
ELSE	2'. $T \rightarrow A: PR_T[H(\text{"cancelled"}, h_A)]$
	T: stores <i>cancel</i> = true

TTP will proceed in one of two possible ways. If the variable *finish* is *true*, it means that *B* or/and *C* had previously finished the protocol with the TTP (see paragraphs below). The TTP had given the NR token to *B* or/and *C*, *ACK_r*, and now it has to give the same NR token to *A*. If *B* and *C* had not contacted with the TTP previously, the TTP will send a message to *A* to *cancel* the transaction, and it will store this information (*cancel* = *true*) in order to satisfy future petitions from *B* or *C*.

If *A* "says" that she has not received message 6 from *C*, *A* may initiate the following *finish* sub-protocol:

	1'. $A \rightarrow T: H(M), h_A, h_B, h_C$
IF (<i>cancel</i> = true)	2'. $T \rightarrow A: PR_T[H(\text{"cancelled"}, h_A)]$
	T: stores <i>cancel_A</i> = true
ELSE	2'. $T \rightarrow A: ACK_T$
	T: stores <i>finish</i> = true

TTP will proceed in one of two possible ways. If the variable *cancel* is *true*, it means that *B* had previously contacted with the TTP (see paragraphs below). The TTP had given a message to *B* to *cancel* the transaction, and now it has to send a similar message to *A*. Additionally, TTP will store the variable *cancel_A* with value *true* to satisfy potential future petitions from *C* (see below). If *B* had not contacted with the TTP previously, the TTP will send the NR token, *ACK_r*, to *A*. In this case the TTP will assign the value *true* to the *finish* variable, in order to satisfy future petitions from *B* or/and *C*.

If *B* "says" that he has not received message 4 from *A*, *B* may initiate the following *cancel* sub-protocol:

	1'. $B \rightarrow T: H(M), h_A, h_B$
IF (<i>finished</i> = true)	2'. $T \rightarrow B: ACK_T$
ELSE	2'. $T \rightarrow B: PR_T[H(\text{"cancelled"}, h_B)]$
	T: stores <i>cancel</i> = true

TTP will proceed in one of two possible ways. If the variable *finished* is *true*, it means that *A* or/and *C* had previously finished the protocol with the TTP (see paragraphs

below and above). The TTP had given the NR token to A or/and C , ACK_T , and now it has to give the same NR token to B . If A and C had not contacted with the TTP previously, the TTP will send a message to B to *cancel* the transaction, and it will store this information (*cancel* = *true*) in order to satisfy future petitions from A or C .

If B “says” that he has not received message 7 from A , B may initiate the following *finish* sub-protocol:

	1'. $B \rightarrow T: H(M), h_A, h_B, h_C, ACK_A$
IF ($cancel_C = true$)	2'. $T \rightarrow B: PR_T[H(“cancelled”, h_B)]$
ELSE	2'. $T \rightarrow B: ACK_T$
	T: stores <i>finish</i> = <i>true</i>

TTP will proceed in one of two possible ways. If the variable *cancel_C* is *true*, it means that A and C (in this order) had previously contacted with the TTP (see paragraph above). The TTP had given a message to A and C to *cancel* the transaction, and now it has to send a similar message to B . If A had not *cancelled* the exchange, the TTP will send the NR token, ACK_T , to B . In this case the TTP will assign the value *true* to the *finish* variable, in order to satisfy future petitions from A or/and C .

If C “says” that he has not received message 5 from B , C may initiate the following *finish* sub-protocol:

	1'. $C \rightarrow T: H(M), h_A, h_B, h_C$
IF ($cancel = true$)	2'. $T \rightarrow C: PR_T[H(“cancelled”, h_C)]$
	T: stores <i>cancel_C</i> = <i>true</i>
ELSE	2'. $T \rightarrow C: ACK_T$
	T: stores <i>finish</i> = <i>true</i>

TTP will proceed in one of two possible ways. If the variable *cancel* is *true*, it means that A or/and B had previously contacted with the TTP (see paragraphs above). The TTP had given a message to A or/and B to *cancel* the transaction, and now it has to send a similar message to C . Additionally, TTP will store the variable *cancel_C* with value *true* to satisfy potential future petitions from B (see above). If A and B had not cancelled the exchange with the TTP previously, the TTP will send the NR token, ACK_T , to C . In this case the TTP will assign the value *true* to the *finished* variable, in order to satisfy future petitions from A or/and B .

If C has not received message 8 from B , C may initiate the following second *finish* sub-protocol:

	1'. $C \rightarrow T: H(M), h_A, h_B, h_C, ACK_A, ACK_B$
IF ($cancel_A = true$)	2'. $T \rightarrow C: PR_T[H(“cancelled”, h_C)]$
ELSE	2'. $T \rightarrow C: ACK_T$
	T: stores <i>finish</i> = <i>true</i>

TTP will proceed in one of two possible ways. If the variable *cancel_A* is *true*, it means that B and A (in this order) had previously contacted with the TTP (see paragraphs above). The TTP had given a message to A and B to *cancel* the transaction, and now it has to send a similar message to C . If A and B had not cancelled the exchange with the TTP previously, the TTP will send the NR token, ACK_T , to C . In this case the TTP will assign the value *true* to the *finished* variable, in order to satisfy future petitions from A or/and B .

In all cases the exchange is conducted to a fair point. The protocol, as in the two-party version, is asynchronous and meets the non-repudiation requirement.

3.2 A N-Party Version

Parties, as in the two-party version, will exchange messages and non-repudiation evidence directly. Only as a last recourse, in the case they cannot get the expected items from the other party, the TTP (T) will be invoked, by initiating *cancel* or *finish* sub-protocols. Messages are over-simplified to stand a clear explanation. The elements used in the protocol description are as follows:

- M message containing the contract to be signed; the contract specifies the order of players in a ring for the exchange
- $h_i = \text{PR}_i[H(M)]$ signature of principal i on the contract M
- $\text{ACK}_i = \text{PR}_i[H(h\text{-ACK})]$ signature of principal i on $h\text{-ACK}$, all the signatures and acknowledgements given by other parties in the ring
- $\text{ACK2}_Y = \text{PR}_Y[H(\text{ACK})]$ last acknowledgement sent to the last player in the protocol
- $\text{ACK}_T = \text{PR}_T[H(M)]$ signature of TTP on $H(M)$; this is an equivalent acknowledgement to which parties should have sent (it is a *NR* evidence)

The *exchange* sub-protocol is as follows:

1. $A \rightarrow B$: M, h_A
2. $B \rightarrow C$: M, h_A, h_B
3. ...
- N. $Z \rightarrow A$: h_B, h_C, \dots, h_Z
- N+1. $A \rightarrow B$: $h_C, \dots, h_Z, \text{ACK}_A$
- N+2. $B \rightarrow C$: $h_D, \dots, h_Z, \text{ACK}_A, \text{ACK}_B$
- N+3. ...
- 2N. $Z \rightarrow A$: $\text{ACK}_B, \text{ACK}_C, \text{ACK}_D, \dots, \text{ACK}_Z$
- 2N+1. $A \rightarrow B$: $\text{ACK}_C, \text{ACK}_D, \dots, \text{ACK}_Z$
- 2N+2. $B \rightarrow C$: $\text{ACK}_D, \dots, \text{ACK}_Z$
- 2N+3. ...
- 3N-1. $Y \rightarrow Z$: ACK2_Y

If a protocol run is completed, everybody will hold non-repudiation (NR) evidence:

- A will hold $h_B, h_C, \dots, h_Z, \text{ACK}_B, \text{ACK}_C, \dots, \text{ACK}_Z$
- B will hold $h_A, h_C, \dots, h_Z, \text{ACK}_A, \text{ACK}_C, \dots, \text{ACK}_Z$
- ...
- Z will hold $h_A, h_B, \dots, h_Y, \text{ACK}_A, \text{ACK}_B, \dots, \text{ACK}_Y, \text{ACK2}_Y$

As you can see $3N-1$ steps are enough if parties behave correctly, in front of BW00 [6] that needs tN^2 (where t is equal or greater than 1). If it is not the case, some of the

parties need to rectify the unfair situation by initiating *cancel* or *finish* sub-protocols, similar to that explained in previous section. First player in the round will have to use similar sub-protocols like *A* in the three-party version. Last player will have to use similar sub-protocols like *C* in the three-party version. All intermediary players will have to use similar sub-protocols like *B* in the three-party version. In the *finish* sub-protocols, TTP may have to send ACK_r to contacting parties.

4 Conclusion

We have presented a fair protocol for N-party contract signing. The fairness is guaranteed provided the existence (and possible involvement) of a trusted third party, that plays a subsidiary role (only intervenes in case of exception). Compared with other solutions, our protocol is doubly efficient: we have reduced the involvement of the TTP and the number of steps in the protocol. The described protocol does not require any deadline to guarantee fairness: each party can contact the TTP when it wants.

References

1. N. Asokan, Matthias Schunter and Michael Waidner: "Optimistic protocols for fair exchange"; Proceedings of 4th ACM Conference on Computer and Communications Security, pages 7-17, Zurich, Switzerland, April 1997.
2. N. Asokan, Victor Shoup and Michael Waidner: "Asynchronous Protocols for Optimistic Fair Exchange"; Proceedings of the IEEE Symposium on Research in Security and Privacy, pages 86-99, Oakland, CA, May 1998.
3. Michael Ben-Or, Oded Goldreich, Silvio Micali and Ronald L. Rivest: "A Fair Protocol for Signing Contracts"; IEEE Transactions on Information Theory, Vol. 36, n. 1, pages 40-46, January 1990.
4. F. Bao, Robert H. Deng and W. Mao: "Efficient and practical fair exchange protocols with off-line TTP"; Proceedings of the IEEE Symposium on Research in Security and Privacy, pages 77-85, Oakland, CA, May 1998.
5. Josep L. Ferrer, Àngel Rotger and Llorenç Huguet: "Firma electrònica de contractos"; Proceedings of III Reunió Espanola de Criptologia, Barcelona (Spain), 1994.
6. Birgit Baum-Weidner and Michael Waidner: "Round-optimal and abuse-free multi-party contract signing"; Proceedings of 27th International Colloquium on Automata, Languages and Programming (ICALP'2000), LNCS 1853, Springer Verlag, pages 524-535, July 2000.
7. Macià Mut, Josep L. Ferrer and Llorenç Huguet: "Certified Electronic Mail Protocol Resistant to a Minority of Malicious Third Parties"; Proceedings of IEEE Infocom 2000, Tel Aviv (Israel), March 2000.
8. Josep L. Ferrer, Llorenç Huguet and Macià Mut: "Protocolo de Correo Electrónico Certificado"; Proceedings of V Reunió Espanola de Criptologia, Málaga, 1998.
9. Ivan Bjerre Damgård: "Practical and provably secure release of a secret and exchange of signatures"; Advances in Cryptology – Proceedings of Eurocrypt'93, LNCS 765, pages 200-217, Springer Verlag, (Lofthus, Norway), May 1993.
10. J.L. Ferrer and L. Huguet: "An Efficient Asynchronous Protocol for Optimistic Certified Electronic Mail"; International Workshop on Cryptographic Techniques & E-commerce, Hong Kong, July 1999.

11. Matthew K. Franklin and Michael K. Reiter: "Fair exchange with a semi-trusted third party"; Proceedings of 4th ACM Conference on Computer and Communications Security, pages 1-6, Zurich, Switzerland, April 1997.
12. N. Asokan, Birgit Baum-Waidner, Matthias Schunter and Michael Waidner: "Optimistic synchronous multi-party contract signing"; Research Report RZ 3089, IBM Research Division, December 1998.
13. Birgit Baum-Waidner and Michael Waidner: "Optimistic asynchronous multi-party contract signing"; Research Report RZ 3078, IBM Research Division, November 1998.
14. N. Asokan, Matthias Schunter and Michael Waidner: "Optimistic protocols for multi-party fair exchange"; Research Report RZ 2892 (#90840), IBM Research, December 1996.
15. T. Okamoto and K. Ohta: "How to simultaneously exchange secrets by general assumptions"; Proceedings of IEEE Symposium on Research in Security and Privacy, pages 14-28, Fairfax, Virginia, November 1994.
16. Shimon Even, Oded Goldreich and Abraham Lempel: "A Randomized Protocol for Signing Contracts"; Communications of the ACM 28/6, pp. 637-647, June 1985.
17. Oded Goldreich: "A simple protocol for signing contracts"; Proceedings of a Workshop on the Theory and Application of Cryptographic Techniques, Crypto '83, Plenum Press, pp. 133-136, New York, 1984.
18. R. Rivest, A. Shamir and L. Adleman: "A Method for Obtaining Digital Signatures and Public Key Cryptosystems"; Communications of the ACM, 21, pages 120-126, 1978.
19. Josep-Lluís Ferrer-Gomila, Magdalena Payeras-Capellà and Llorenç Huguet i Rotger: "An Efficient Protocol for Certified Electronic Mail"; Proceedings of Third International Workshop on Information Security, ISW 2000, LNCS 1751, Springer Verlag, pages 237-248, Wollongong, Australia, December 2000.
20. Jianying Zhou and Dieter Gollmann: "An Efficient Non-repudiation Protocol"; Proceedings of 10th IEEE Computer Security Foundations Workshop, pages 126-132, Rockport, Massachusetts, IEEE Computer Society Press, June 1997.
21. Jianying Zhou, Robert Deng and Feng Bao: "Some Remarks on a Fair Exchange Protocol"; Proceedings of Third International Workshop on Practice and Theory in Public Key Cryptosystems, PKC 2000, LNCS 1751, Springer Verlag, pages 46-57, Melbourne, Victoria, Australia, January 2000.
22. Juan A. Garay, Markus Jakobsson and Philip MacKenzie: "Abuse-Free Optimistic Contract Signing"; Advances in Cryptology – Proceedings of CRYPTO'99, LNCS-1666, pages 449-466, 1999.
23. Birgit Pfitzmann, Matthias Schunter and Michael Waidner: "Optimal Efficiency of Optimistic Contract Signing"; Proceedings of 17th Symposium on Principles of Distributed Computing, PODC'98, ACM, New York, pages 113-122, 1998.

Appendix 1: Verifiability of TTP in the Two-Party Protocol

The protocol satisfies the verifiability requirement, if the TTP can be forced to send a valid response to any request send to it. A TTP's possible misbehavior is: A receives h_B and h_B' , while B receives the *cancel* token. If A uses h_B and h_B' to prove that B has signed the contract, B can use the *cancel* token to prove the TTP's misbehavior. If A received h_B and then (or before) she cancelled the exchange (and so she did not receive h_B'), and try to use h_B to prove that B received the evidence, and the TTP has the h_{AT} token, it is clear A 's misbehavior.

The other TTP's possible misbehavior is: B receives ACK_T while A receives the *cancel* token. If B uses h_A and ACK_T to prove that A has signed the contract, A can use the *cancel* token to prove the TTP's misbehavior. It should be noted that if B uses h_A

and ACK_A to prove that A has signed the contract, A can not use the *cancel* token to prove the TTP's misbehavior, since the TTP did not issue conflicting evidence (clearly A is misbehaving).

Appendix 2: Analysis of Three-Party Protocol

It is not obvious that no party can achieve an advantage in any stage of a protocol run. Here we will analyze one situation parties can face (all other situations can be, and have been, analyzed in a similar way). It has to be fixed that TTP always acts in an atomic way: it processes a petition after other petition (in petitions related to the same exchange). A general principle is that all parties know they have to contact with TTP after they have sent one message, if they do not receive all evidence from other parties. Of course a cheating party can contact the TTP not as thought in the described protocol, but even in this case, the proposed protocol is fair.

The case we are going to analyze is an execution of the protocol stopped after step 6 (and before step 7). We will have to handle some special conflicting situations: A already has *NR* evidence.

- C contacts with TTP before A and B . C will obtain a *NR* token, and after this, A and B will not be able to *cancel* the exchange. A and B can run their *cancel* or *finish* sub-protocols, but in both cases they will obtain a *NR* token.
- B contacts with TTP before A and C , to *finish* the exchange. B will obtain a *NR* token, and after this, A will not be able to *cancel* the exchange. And if A or C contact with the TTP, they will obtain a *NR* token.
- B runs *cancel* sub-protocol, A runs *cancel* sub-protocol and C runs his first *finish* sub-protocol (in this order). B will obtain a *cancel* message. Then, A contacts with TTP to *cancel* the exchange. TTP checks *cancel* and observes that it is *true*, and TTP sends a *cancel* message to A . After this, if C contacts with TTP, with the first *finish* sub-protocol, TTP will check *cancel*, and the value is *true*. So, it will send a *cancel* message to C . All parties have been lying (sending less evidence that they have), but the protocol ends in a fair way (all of them with a *cancel* message).
- B runs *cancel* sub-protocol, A runs *cancel* sub-protocol and C runs his second *finish* sub-protocol (in this order). B will obtain a *cancel* message. Then, A contacts with TTP to *cancel* the exchange. TTP checks *finish* and observes that it is *false*, and TTP sends a *cancel* message to A . After this, if C contacts with TTP, with the second *finish* sub-protocol, TTP will check *cancel_A*, and the value is *false*. Now the TTP can see that A and B have been cheating: ACK_B proves that B had received ACK_A and A had received h_C , and so they were able to *finish* (and not to *cancel*) the exchange. So, TTP will send *NR* evidence to C . B does not have *NR* evidence, but it is a fault of him (not of the protocol). If A or B try to use the *cancel* messages in courts against C , *NR* token of C will show that they are lying.
- B runs *cancel* sub-protocol, A runs *finish* sub-protocol and C runs his first *finish* sub-protocol (in this order). B will obtain a *cancel* message. Then, A contacts with TTP to *cancel* the exchange. TTP checks *cancel* and observes that it is *true*, and TTP sends a *cancel* message to A . After this, if C contacts with TTP, with the first *finish* sub-protocol, TTP will check *cancel*, and the value is *true*. So, it will send a *cancel* message to C . All parties have been lying (sending less evidence that they

have), but the protocol ends in a fair way (all of them with a *cancel* message). *A* can not use *NR* evidence, because it will prove she was cheating.

- *B* runs *cancel* sub-protocol, *A* runs *finish* sub-protocol and *C* runs his second *finish* sub-protocol (in this order). *B* will obtain a *cancel* message. Then, *A* contacts with TTP to *finish* the exchange. TTP checks *cancel* and observes that it is *true*, and TTP sends a *cancel* message to *A*, and sets $cancel_A$ to value *true*. After this, if *C* contacts with TTP, with the second *finish* sub-protocol, TTP will check $cancel_A$, and the value is *true*. Now the TTP can see that *B* have been cheating: ACK_B proves that *B* had received ACK_A , but TTP did not know it when it sent a *cancel* message to *A* (TTP could think that *A* had followed the protocol). And so, TTP will send a *cancel* message to *C*. Now, *A* has *NR* evidence and a *cancel* message. But, if *A* tries to use the *NR* evidence she possesses to declare the contract is signed, *C* will show that she is lying: $cancel_A$ proves *A* "tried" falsely to *finish* the protocol and this exchange was *cancelled*.
- *B* runs *cancel* sub-protocol, *C* runs his first *finish* sub-protocol, and *A* does not contact with TTP (in this order). *B* and *C* will obtain a *cancel* message. For the TTP this situation is coherent: it thinks that the exchange stopped before step 4. Both have been lying (they have more evidence), and *A* has enough evidence to prove the contract is signed, and that *B* and *C* have been cheating.
- *B* runs *cancel* sub-protocol, *C* runs his first *finish* sub-protocol, and *A* runs *cancel* or *finish* sub-protocols (in this order). *B* will obtain a *cancel* message, and after this, *C* and *A* only will obtain a *cancel* message. For the TTP this situation is coherent: it thinks that the exchange stopped before step 4. All of them have been lying (they have more evidence). *A* has evidence to prove the contract is signed, and a *cancel* message, but it is because *B* and *C* have been cheating.
- *B* runs *cancel* sub-protocol, *C* runs his second *finish* sub-protocol, and *A* does not contact TTP or runs *cancel* or *finish* sub-protocols (in this order). *B* will obtain a *cancel* message. Then, if *C* contacts with TTP, with the second *finish* sub-protocol, TTP will check $cancel_A$, and the value is *false*. Now the TTP can see that *B* has been cheating: ACK_B proves that *B* had received ACK_A , and so he was able to *finish* (and not to *cancel*) the exchange. So, TTP will send *NR* evidence to *C*, and sets *finish* to *true* and *cancel* to *false*. After this, if *A* contacts with TTP, to *cancel* or *finish* the protocol run, TTP will check *finish* or *cancel*, respectively, and in one case the value is *true* and in the other *false*. So, it will send a *NR* token to *A*. Whatever case, *A* already has sufficient evidence to prove the contract is signed, and generally she will not contact TTP. Only *B* does not have *NR* evidence, but it is a fault of him (not of the protocol). If he tries to use the *cancel* message in courts against *C* or *A*, *NR* tokens of *C* and *A* will show that he is lying.
- *A* contacts with TTP before *B* and *C*, to *finish* the exchange (she sends h_A , h_B and h_C). *A* will obtain a *NR* token, and so, after this, *B* and *C* will obtain a *NR* token from the TTP. *B* can run the *cancel* or *finish* sub-protocols, but in both cases TTP will send a *NR* token. You can see that it is a nonsense: *A* only achieves the same she already had.
- *A* runs *cancel* sub-protocol, *B* runs *cancel* sub-protocol and *C* runs his first *finish* sub-protocol (in this order). *A* will obtain a *cancel* message, and after this, *B* and *C* only will obtain a *cancel* message. For the TTP this situation is coherent: it thinks that the exchange stopped before step 4. *A* and *B* have been lying (they have more

evidence). Besides, only A has evidence to prove the contract is signed, and a *cancel* message, but it is because B and C have been cheating.

- A runs *cancel* sub-protocol, B runs *cancel* sub-protocol and C runs his second *finish* sub-protocol (in this order). B will obtain a *cancel* message. Then, A contacts with TTP to *cancel* the exchange. TTP checks *finish* and observes that it is *false*, and TTP sends a *cancel* message to A . After this, if C contacts with TTP, with the second *finish* sub-protocol, TTP will check $cancel_A$, and the value is *false*. Now the TTP can see that A and B have been cheating: ACK_B proves that B had received ACK_A and A had received h_C , and so they were able to *finish* (and not to *cancel*) the exchange. So, TTP will send *NR* evidence to C . A and B do not have *NR* evidence, but it is a fault of them (not of the protocol). If they try to use the *cancel* messages in courts against C , *NR* token of C will show that they are lying. Only B does not have *NR* evidence, but it is a fault of him (not of the protocol). If he tries to use the *cancel* message in courts against C or A , *NR* tokens of C or A will show that he is lying. Besides, A has evidence to prove the contract is signed, and a *cancel* message, but she can not use the *cancel* message because C can show the *NR* evidence to prove she is lying.
- A runs *cancel* sub-protocol, B runs *finish* sub-protocol and C runs some of the *finish* sub-protocols (in this order). A will obtain a *cancel* message. Then, B contacts with TTP to *finish* the exchange. TTP checks $cancel_C$ and observes that it is *false*, and TTP deducts that A was cheating, and so sends a *NR* evidence to B . After this, C contacts with TTP, and TTP will send *NR* evidence to C . All parties have *NR* evidence (even A). A has also a *cancel* message that can not be used in courts, because it proves that A is a cheating party.
- A runs *cancel* sub-protocol, C runs his first *finish* sub-protocol, and B runs *cancel* or *finish* sub-protocols (in this order). A will obtain a *cancel* message. Then, C contacts with TTP to *finish* the exchange. TTP checks *cancel* and observes that it is *true*, and TTP sends a *cancel* message to C , and sets variable $cancel_C$ to value *true*. After this, if B contacts with TTP, to *cancel* or *finish* the protocol run, TTP will check *cancel* or $cancel_C$, respectively, and in both cases the value is *true*. So, it will send a *cancel* message to B . Some parties have been lying (sending less evidence that they have), but the protocol ends in a fair way (all of them with a *cancel* message). A has also *NR* evidence that can not be used in courts, because it proves that A is a cheating party.
- A runs *cancel* sub-protocol, C runs his second *finish* sub-protocol, and B runs *cancel* or *finish* sub-protocols (in this order). A will obtain a *cancel* message. Then, if C contacts with TTP, with the second *finish* sub-protocol, TTP will check $cancel_A$, and the value is *false*. Now the TTP can see that A has been cheating: ACK_A proves that A had received h_C , and so she was able to *finish* (and not to *cancel*) the exchange. So, TTP will send *NR* evidence to C , and sets *finish* to *true* and *cancel* to *false*. After this, if B contacts with TTP, to *cancel* or *finish* the protocol run, TTP will check *finish* or $cancel_C$, respectively, and in one case the value is *true* and in the other *false*. So, it will send a *NR* token to B . At the end all of them have *NR* evidence. Besides, A has a *cancel* message, but she can not use the *cancel* message because C and B can show the *NR* evidence to prove she is lying.

As a conclusion, it can be proved that if parties follow the protocol, they always will be in a fair position.

Efficient Sealed-Bid Auctions for Massive Numbers of Bidders with Lump Comparison

Koji Chida, Kunio Kobayashi, and Hikaru Morita

NTT Information Sharing Platform Laboratories, NTT Corporation
1-1 Hikarinooka, Yokosuka-shi, Kanagawa 239-0847, Japan
{chida,kunio,morita}@isl.ntt.co.jp

Abstract. A new scheme for electronic sealed-bid auctions that preserves losing bids is presented. By this scheme, the computational complexity of the opening phase can be reduced to $O(\log \ell)$; previous works required $O(N \cdot \ell)$ or $O(N \cdot \log \ell)$ where the number of bidders is N and the range of bids is ℓ . The proposed scheme has two technical points. One is that computational complexity is independent of the number of bidders. The other is a new efficient *value-comparing* method. These techniques allow our auction scheme to be more than five hundred times faster than previous schemes. Furthermore, our auction scheme can be eleven million times faster than previous schemes if it is assured that auctioneers do not conspire.

1 Introduction

Kikuchi et al. [KHT99] proposed an interesting sealed-bid auction that preserves the privacy of losing bids, i.e. anyone cannot know the losing bids [1]. It is equivalent to Dutch auctions, where the value used to compare the bids is gradually reduced until it reaches the first successful bid.

One of the most interesting themes for auctions is how to reduce the computational complexity of determining the successful bid. Those of previous works are either $O(N \cdot \ell)$ or $(N \cdot \log \ell)$ where the number of bidders is N and the range of bids is ℓ . If N is large, they incur huge computational complexity. This means that they cannot be applied to large-scale auctions.

This paper presents a new auction scheme. A strong point of the proposed scheme is that computational complexity is $O(\log \ell)$, i.e. independent of N . The proposed scheme has two technical points.

One is that computational complexity is independent of the number of bidders. Previous works examined each bid. In the proposed scheme, however, each bidder passes a different piece of her/his bid to two auctioneers. Auctioneers bundle all bidders' pieces and compare them later.

The other is a new efficient *value-comparing* method. It allows two auctioneers to judge whether two values are equal or not without leaking them. In our auction schemes, *value-comparing* is needed and its performance affects the performance of the entire auction. Naor and Pinkas proposed a similar method as

¹ This type of auction is called *auction* hereafter.

one of application using OPE(Oblivious Polynomial Evaluation)[\[NP99\]](#). Their method uses many exponentiations. This paper proposes the efficient value-comparing methods, whose security are based on the modified DDH(Decision Diffie-Hellman) assumption.

Due to the above mentioned techniques, the proposed scheme is more than five hundred times faster than previous schemes, can be applied to large-scale auctions, and is practical.

This paper is organized as follows. Section [2](#) defines notations used in this paper. Section [3](#) introduces previous works. Section [4](#) describes our new auction scheme. Sections [5](#) and [6](#) discuss the security and computational complexity of our new auction scheme. The paper concludes in Sect. [7](#)

2 Notation

The following notation is used throughout this paper.

- A, B : auctioneers
- i : the i -th bidder ($i = 1, \dots, N$)
- BB : a bulletin board
- $V = \{v_1, \dots, v_\ell\}$: a set of bids ($v_j < v_{j+1}$)
- $h(\cdot)$: a practical one-way hash function (e.g., SHA-1)
- t, t' : security parameters
- $a \parallel b$: concatenation of a and b
- $\lfloor x \rfloor$: the greatest integer equal to or less than x

3 Previous Works

[\[KHT99\]](#) showed the necessity of preserving the privacy of losing bids and presented an auction scheme that uses a distributed decryption technique. [\[NPS99\]](#) created a scheme by combining Yao’s secure computation with oblivious transfer. [\[Cac99\]](#) proposed a scheme using homomorphic encryption and oblivious third party. [\[Sak00\]](#) proposed a scheme in which a bid is represented by an encrypted message whose public key corresponds to her/his bidding price.

Table [1](#) shows the auctioneers’ computational complexity of each previous work.

Table 1. Computational complexity of previous works

Reference	Computational complexity
[KHT99]	$O(N \cdot \ell)$
[NPS99]	$O(N \cdot \log \ell)$
[Cac99]	$O(N \cdot \log \ell)$
[Sak00]	$O(N \cdot \ell)$

In this section, [Sak00] is mentioned as an example of $O(N \cdot \ell)$ scheme. Similarly, [Cac99] is taken as an example of $O(N \cdot \log \ell)$ schemes.

3.1 [Sak00]

Let N bidders and two auctioneers A, B exist, and suppose the largest bid is the successful one.

[Set-up.]

- The auctioneers determine a set of encryption functions $\{E_{v_j}\}$, decryption functions $\{D_{v_j}\}$ and messages $\{M_{v_j}\}$ for $v_j \in V$ ($j = 1, 2, \dots, \ell$).

[Bidding.]

- Bidder i chooses v_{d_i} from V as her/his bidding price.
- She/He bids $E_{v_{d_i}}(M_{v_{d_i}})(= C_{B_i})$ with her/his signature.

[Opening.] The auctioneers

- set $k \leftarrow \ell$ and decrypt distributively all C_{B_i} s with D_{v_k} for all i .
- while $D_{v_k}(C_{B_i}) \neq M_{v_k}$ for all i , set $k \leftarrow k - 1$ and return to previous step.
- publish v_k as the winning price.

3.2 [Cac99]

Let N bidders and two auctioneers A, B exist, and suppose the largest bid is the successful one.

[Set-up.]

- Bidder i chooses v_{d_i} from V as her/his bidding price.
- Bidders and A are connected to B by secure channels.
- Let $E_k: \{0, 1\}^k \times \mathcal{X} \rightarrow \mathcal{C}$ denote the public encryption function for a message x using a random string u , and let $D_k: \mathcal{C} \rightarrow \mathcal{X}$ be the corresponding secret decryption function. A generates E_k, D_k and publishes E_k .
- B chooses the following random values for $\gamma \in [1, N]$,
 - $\kappa_\gamma \in \{0, 1\}^k, x_{\gamma,c}, x_{\gamma,w}, s_{\gamma,w} \in \mathcal{X}$ for $w \in [0, c-1]$, where c be the integer such that $2^{c-1} \leq v_\ell < 2^c$,
 - $t_\gamma^A, t_\gamma^B \in \{0, 1\}^k$,
 - p_γ^A, p_γ^B : primes, m_γ^A, m_γ^B ($p_\gamma^A \mid \varphi(m_\gamma^A), p_\gamma^B \mid \varphi(m_\gamma^B)$).

[Bidding.]

- Let $H_k: \{0, 1\}^k \times \{0, 1\}^* \rightarrow \{0, 1\}^{k'}$, and B determines $\delta_{\gamma,w}^A = H_k(\kappa_\gamma \parallel x_{\gamma,w} + s_{\gamma,w}) \oplus t_\gamma^A$, $\delta_{\gamma,w}^B = H_k(\kappa_\gamma \parallel x_{\gamma,w} - s_{\gamma,w}) \oplus t_\gamma^B$ and sends $(x_{\gamma,c}, x_{\gamma,w}, s_{\gamma,w}, \delta_{\gamma,w}^A, \delta_{\gamma,w}^B)$ to all bidders by the secure channels.
- Bidder i chooses randomly $u_{i,\gamma,w}, u'_{i,\gamma,w} \in \{0, 1\}^k$, computes the following for $\gamma \in [1, N]$;

$$y_{i,\gamma,w}^A = \begin{cases} E_k(u_{i,\gamma,w} \parallel x_{\gamma,w} - x_{\gamma,w+1}), & \text{if } v_{d_i,w} = 0 \\ E_k(u_{i,\gamma,w} \parallel x_{\gamma,w} - x_{\gamma,w+1} + s_{\gamma,w}), & \text{if } v_{d_i,w} = 1 \end{cases}$$

$$y_{i,\gamma,w}^B = \begin{cases} E_k(u'_{i,\gamma,w} \parallel 0), & \text{if } v_{d_i,w} = 0 \\ E_k(u'_{i,\gamma,w} \parallel -s_{\gamma,w}), & \text{if } v_{d_i,w} = 1 \end{cases}$$

($v_{d_i} = (v_{d_i,c-1} v_{d_i,c-2} \cdots v_{d_i,0})_2$), and sends $y_{i,\gamma,w}^A, y_{i,\gamma,w}^B$ to B .

[Opening.]

- B chooses a random permutation π_1, \dots, π_N of $[1, N]$ and lets $\gamma_{max} = 1$. A and B repeat the following Step- $\{1, 2, 3, 4\}$ for $\gamma = 2, \dots, N$.
- Step-1: B computes $z_{\gamma,w} = y_{\pi_\gamma, \gamma, w}^A \cdot y_{\pi_{\gamma_{max}}, \gamma, w}^B$ and sends $x_{\gamma,c}, z_{\gamma,w}, \delta_{\gamma,w}^A, \delta_{\gamma,w}^B, m_\gamma^A, m_\gamma^B$ to A for $w \in [0, c-1]$.
- Step-2: A lets $\alpha_{\gamma,c} = x_{\gamma,c}$ and chooses $g_{A,c} \in QR_{m^A}$ and $g_{B,c} \in QR_{m^B}$.
- Step-3: A performs the following for $w = c-1, \dots, 0$:
- $\alpha_{\gamma,w} = \alpha_{\gamma,w+1} + D_k(z_{\gamma,w})$;
 $q_{A,w} = \lambda(H_k(\kappa_\gamma \parallel \alpha_{\gamma,w}) \oplus \delta_{\gamma,w}^A)$;
 $g_{A,w} = (g_{A,w+1})^{q_{A,w}} \bmod m_\gamma^A$;
 $q_{B,w} = \lambda(H_k(\kappa_\gamma \parallel \alpha_{\gamma,w}) \oplus \delta_{\gamma,w}^B)$;
 $g_{B,w} = (g_{B,w+1})^{q_{B,w}} \bmod m_\gamma^B$;
 $(\lambda \text{ is a map that associates a } k'\text{-bit prime } p \text{ with } k'\text{-bit string } x.)$
 - chooses $r_A \in \mathbb{Z}/m^A\mathbb{Z}$ and $r_B \in \mathbb{Z}/m^B\mathbb{Z}$ randomly and independently, computes $h_\gamma^A = (g_{A,0})^{r_A}$, $h_\gamma^B = (g_{B,0})^{r_B}$.
 - sends h_γ^A, h_γ^B to B .
- Step-4: B determines the winner (π_γ or $\pi_{\gamma_{max}}$) by testing whether h_γ^A has a p_γ^A -th root. If and only if the test succeeds does B set γ_{max} to γ .
- B outputs $\pi_{\gamma_{max}}$ and declares that the successful bidder is γ_{max} .

4 Our Scheme

If N is large, the previous works incur huge computational complexity. Therefore, they cannot be applied to large-scale auctions. N has no influence on the computational complexity of the proposed scheme, so it supports large-scale auctions.

This section introduces the new scheme. We then discuss the case where the smallest bid is the successful one. Finding the largest bid is performed in essentially the same manner.

4.1 Bidding Phase

Every bidder i chooses a bid $v_{d_i} \in V$. She/He generates $a_i[x], b_i[x] \in \{0, 1\}^t$ ($x = 1, 2, \dots, \ell$) such that

$$a_i[x] = b_i[x] \quad \text{if } d_i > x$$

$$a_i[x] \neq b_i[x] \quad \text{otherwise}$$

and chooses $r_{i,A}, r_{i,B} \in_R \{0, 1\}^{t'}$.

She/He sends $h(r_{i,A} \parallel a_i)$ and $h(r_{i,B} \parallel b_i)$ ($a_i = \{a_i[x]\}, b_i = \{b_i[x]\}$) to BB as bit commitments, and sends $(r_{i,A}, a_i)$ to A and $(r_{i,B}, b_i)$ to B by secure channels.

4.2 Opening Phase

Auctioneers perform following either *Selection of successful bid-1* or *Selection of successful bid-2* to determine a successful bid. Selection of successful bid-1 is practical and can be performed efficiently. However, if the successful bidder conspires with any of the auctioneers, the second smallest bid, a losing bid, may be revealed. Selection of successful bid-2 solves this problem though its computation complexity is larger than Selection of successful bid-1.

Selection of successful bid-1

0. Set $max \leftarrow \ell$, $min \leftarrow 0$.
1. Set $w \leftarrow \lfloor (max + min)/2 \rfloor$.
2. Auctioneers compute $h(c_{w,A})$ and $h(c_{w,B})$, where $c_{w,A} = a_1[w] \parallel a_2[w] \parallel \dots \parallel a_N[w]$ and $c_{w,B} = b_1[w] \parallel b_2[w] \parallel \dots \parallel b_N[w]$. They send $h(c_{w,A}), h(c_{w,B})$ to BB (by this operation, you can judge whether the successful bid v_{min} is greater than v_w . For example, the case $h(c_{w,A}) \neq h(c_{w,B})$ means that at least one bid is equal to or less than v_w).
3. If $v_{min} > v_w$, set $min \leftarrow w + 1$. Otherwise, set $max \leftarrow w$.
4. If $min = max$, output v_{min} as the successful bid and terminate the procedure. Otherwise, go to 1.

By one-time comparing $h(c_{w,A})$ to $h(c_{w,B})$, the number of candidates is nearly halved. Therefore, we can determine the successful bid by $\log \ell$ -times comparisons.

After the successful bid is determined, A, B send $c_{min,A}, c_{min,B}$ to BB . By comparing $c_{min,A}$ to $c_{min,B}$, anyone can identify the successful bidder z . Note that, $c_{min,A}, c_{min,B}$ published by auctioneers are assured by $h(c_{min,A}), h(c_{min,B})$, and anyone can identify the successful bidders even if multiple successful bidders exist. Moreover, A, B send $a_z, r_{z,A}, b_z, r_{z,B}$ to BB . By these and $h(r_{z,A} \parallel a_z), h(r_{z,B} \parallel b_z)$, which are bit commitments, anyone can verify a_z, b_z .

In successful bid-1, if the successful bidder conspires with any of the auctioneers, the second smallest bid may be revealed, because $h(c_{w,A}), h(c_{w,B})$ give significant information. For example, the successful bidder z has sent a_z, b_z to A, B , respectively. If z reveals b_z to A , A may be able to determine the second smallest bid by computing

$$c'_{w,A} = a_1[w] || a_2[w] || \cdots || b_z[w] || \cdots || a_N[w].$$

We can prevent this dishonest act with the *Selection of successful bid-2*.

Selection of successful bid-2 ²

Let G be a finite cyclic group of order p where p is a large prime, and π be a function such that $\pi: \{0, 1\}^* \rightarrow G$.

0. Set $max \leftarrow \ell$, $min \leftarrow 0$.
1. Set $w \leftarrow \lfloor (max + min)/2 \rfloor$.
2. A chooses $s_{w,A} \in_R \mathbb{Z}/p\mathbb{Z}$, computes $\pi(c_{w,A})^{s_{w,A}}$ and sends it with bit commitment $h(s_{w,A} || c_{w,A})$ to BB .
3. B chooses $s_{w,B} \in_R \mathbb{Z}/p\mathbb{Z}$, computes $\pi(c_{w,B})^{s_{w,B}}$ and sends it with bit commitment $h(s_{w,B} || c_{w,B})$ to BB .
4. A computes $(\pi(c_{w,B})^{s_{w,B}})^{s_{w,A}}$ and sends it to BB .
5. B computes $(\pi(c_{w,A})^{s_{w,A}})^{s_{w,B}}$ and sends it to BB (operations-4,5 allow us to judge whether v_{min} is greater than v_w).
6. If $v_{min} > v_w$, set $min \leftarrow w + 1$. Otherwise, set $max \leftarrow w$.
7. If $min = max$, output v_{min} as the successful bid and terminate the procedure. Otherwise, go to 1.

The one-time comparison of $(\pi(c_{w,B})^{s_{w,B}})^{s_{w,A}}$ to $(\pi(c_{w,A})^{s_{w,A}})^{s_{w,B}}$ nearly halves the number of candidates. Therefore, we can determine the successful bid by $\log \ell$ -times comparisons.

After the successful bid is determined, A, B send $(s_{w,A}, c_{w,A}), (s_{w,B}, c_{w,B})$ to BB . Anyone can identify the successful bidder z by using these values. Note that, anyone can identify the successful bidders, even if multiple successful bidders exist. Moreover, A, B send $a_z, r_{z,A}, b_z, r_{z,B}$ to BB . By these and $h(r_{z,A} || a_z), h(r_{z,B} || b_z)$, which are bit commitments, anyone can verify a_z, b_z .

In the opening phase, A, B bundle all bidders' a_i, b_i into $c_{w,A}, c_{w,B}$ respectively. The Selection of successful bid- $\{1,2\}$ uses $c_{w,A}, c_{w,B}$, not a_i, b_i . Therefore, N has no influence on the computational complexity of the proposed scheme.

If all bids are greater than v_w , $c_{w,A} = c_{w,B}$ holds. If any bid is equal to or less than v_w , $c_{w,A} \neq c_{w,B}$. If you know $c_{w,A}$ and $c_{w,B}$ ($c_{w,A} \neq c_{w,B}$), you can narrow down the candidates for a successful bidder. Thus, the privacy of bidders is lost. The Selection of successful bid-2 solves this problem by using *value-comparing*. This is described in the appendix.

² If this selection is used, security parameter t can be set to 1 in the bidding phase.

5 Security

This section considers the security of our scheme. Three properties of restricted auctions: *fairness*, *verifiability* and *privacy of losing bids* are defined below:

- Fairness: Nobody can know the bids before the opening phase.
- Verifiability: Everybody can verify the behavior of the auctioneers and the successful bidder.
- Privacy of losing bids: Nobody can know the losing bids even after the auction.

Fairness

The data published by bidders before the opening phase is just $h(r_{i,A}||a_i)$ and $h(r_{i,B}||b_i)$ (see Sect. 4.1). Finding both a_i and b_i is equivalent to finding bid v_{d_i} , but a_i and b_i are sent by secure channels. Of course, just a_i or b_i gives no information. Also, finding a_i and b_i by $h(r_{i,A}||a_i)$ and $h(r_{i,B}||b_i)$ can be reduced to the one-wayness of a practical hash function.

Verifiability

Verifiability is based on the bit commitments of the successful bidder z and auctioneers, namely,

$$\begin{aligned}
 V_1 &= \{h(r_{z,A}||a_z), h(r_{z,B}||b_z)\}, \\
 V_{2,1} &= \{h(c_{min,A}), h(c_{min,B})\}, \\
 V_{2,2} &= \{h(s_{min,A}||c_{min,A}), h(s_{min,B}||c_{min,B})\}, \\
 V_{3,1} &= \{c_{min,A}, c_{min,B}\}, \\
 V_{3,2} &= \{s_{min,A}, c_{min,A}, s_{min,B}, c_{min,B}\} \text{ and} \\
 V_4 &= \{a_z, r_{z,A}, b_z, r_{z,B}\} \\
 &\quad \text{(see Sects. 4.1 and 4.2).}
 \end{aligned}$$

Here $V_{*,1}$ means that the case of using Selection of successful bid-1. Similarly, $V_{*,2}$ means the case of using Selection of successful bid-2. The correctness of bit commitments V_1 and $V_{2,*}$ can be verified by $V_{3,*}$ and V_4 .

Selection of successful bid- $\{1,2\}$ may be able to identify a successful bid when any bidder or auctioneer performs the bidding/opening phase incorrectly. Examples include the case that an auctioneer performs the opening phase incorrectly and v_{min} is wrong. However, bidders who committed $v_{d_i} < v_{min}$ can prove the auctioneer's fraud by publishing her/his secret value $\{a_i, r_{i,A}, b_i, r_{i,B}\}$. As the other case, bidder k that sends random values to disturb an auction, the following three cases are considered.

Suppose that v_{min} is the smallest bid among all *correct* bidders, and bidder k generates a_k and b_k randomly without any rule.

The case of $min < y$ proceeds as follows (where y is a minimum value such that $a_k[y] \neq b_k[y]$): (I) *The bidder k does not become the successful bidder.* (II)

There is no effect on the opening phase. (III) Her/His cheating is not made public.

The case of $\min = y$ proceeds as follows: Although she/he becomes the successful bidder temporarily, her/his cheating is made public after auctioneers publish her/his random commitment.

The case of $\min > y$ yields the same result as either the case of $\min < y$ or $\min = y$.

Therefore, her/his cheating is made public if and only if she/he becomes the successful bidder temporarily. Once discovered, the auction can be repeated for the bidders but excluding the forger.

Privacy of losing bids

By using Selection of successful bid-2, losing bids are kept secret under the modified DDH assumption. This is discussed in the appendix.

6 Computational Complexity

The computational complexity of the opening phase is discussed in this section since the load factor of this phase is the highest of all phases. Table 2 shows the computational complexity of our methods, [Cac99] and [Sak00]. C_{hash} and C_{me} denote the computational complexity of a hash function and 1024-bit modular exponentiation, respectively. ω in [Sak00] denotes the number of auctioneers. *Our method*-{1,2} denotes our scheme using Selection of successful bid-{1,2}, respectively.

The ratios in Table 2 denote the computational complexity of each method with the following conditions:

- The ratio of C_{hash} to C_{me} is 1:10000 [RS96].
- We set $\omega = 2$ to match the condition used in [Cac99].
- $N = 1000$, $\ell = 2^{10}$.
- Set computational complexity of method-1 to 1.

The computational complexities of our methods are less than one five hundredth that of [Cac99]. Note that method-1 may leak bits of information about losing bids if the successful bidder conspires with either auctioneer. However, if the successful bidder does not, method-1 is very practical.

7 Conclusion

This paper proposed a new approach to auction schemes for massive numbers of bidders. Our new scheme has two new technical points. One is bundling all bidders' bids and lump comparison. This makes the computational complexity independent of the number of bidders. The other is a new efficient *value-comparing* method. These techniques allow our auction scheme(method-1) to

Table 2. Computational complexity of the opening phase

	computational complexity	ratio
Our method-1	$2 \log \ell \times C_{hash}$	1
Our method-2	$2 \log \ell \times (C_{hash} + 2C_{me})$	20,000
Cac99	$(N - 1) \log \ell \times (2C_{hash} + \frac{7}{3}C_{me})$	11,667,500
Sak00	$\frac{\omega N \ell}{2} \times C_{me}$	500,000,000

be more than five hundred times faster than previous schemes. Furthermore, our auction scheme(method-2) can be eleven million times faster than previous schemes if it is assured that auctioneers do not conspire. A future subject is examination and cutting down the amount of communications.

References

- [Bon98] D. Boneh, "The Decision Diffie-Hellman Problem," Proc. of ANTS-III, LNCS 1423, Springer-Verlag, pp. 48-63, 1998.
- [Cac99] C. Cachin, "Efficient Private Bidding and Auctions with an Oblivious Third Party," Proc. of ACM-CCS'99, 1999.
- [DH76] W. Diffie and M. Hellman, "New Directions in Cryptography," IEEE Trans. on Information Theory, IT-22, 6, pp. 644-654, 1976.
- [JY96] M. Jakobsson, M. Yung, "Proving Without Knowing: On Oblivious, Agnostic and Blindfolded Provers," Proc. of CRYPTO 96, LNCS 1109, Springer-Verlag, pp. pp. 186-200, 1996.
- [KHT99] H. Kikuchi, M. Harkavy and D. Tygar, "Multi-Round Anonymous Auction Protocols," IEICE Trans. Inf. & Syst., Vol. E82-D, No.4, pp. 769-777, 1999.
- [NP99] M. Naor and B. Pinkas, "Oblivious Transfer and Polynomial Evaluation," Proc. of 31st STOC, pp. 245-254, 1999.
- [NPS99] M. Naor, B. Pinkas and R. Sumner, "Privacy Preserving Auctions and Mechanism Design," ACM Workshop on E-Commerce, 1999.
- [NR97] M. Naor and O. Reingold, "Number-Theoretic Constructions of Efficient Pseudo-Random Functions," Extended abstract in: Proc. of 38th FOCS, pp. 458-467, 1997.
- [RS96] R. L. Rivest and A. Shamir, "PayWord and MicroMint—Two simple micropayment schemes," Proc. of 1996 International Workshop on Security Protocols, LNCS 1189, pp. 69-87, 1996.
- [Sak00] K. Sako, "An auction protocol which hides bids of losers," Proc. of PKC'2000, pp. 422-432, 2000.
- [Yao82] A.C. Yao, "Protocols for Secure Computations," Proc. of FOCS, pp. 160-164, 1982.
- [Yao86] A.C. Yao, "How to Generate and Exchange Secrets," Proc. of FOCS, pp. 162-167, 1986.

Appendix

A Value-Comparing

This section discusses the following problem:

Problem 1. Let $\mathcal{A} = \{\alpha_0, \alpha_1, \alpha_2\}$ be a set of three elements.³ Suppose that Alice and Bob choose α_i, α_j in \mathcal{A} , respectively. They would like to check whether α_i equals α_j without leaking them.

Here we prove that $c_{w,A}$ and $c_{w,B}$ are kept secret in Selection of successful bid-2 even if the successful bidder conspires with any of the auctioneers. We already noted that Selection of successful bid-1 leaks $h(c_{w,A})$ and $h(c_{w,B})$ in the comparison $c_{w,A}$ and $c_{w,B}$, and $h(c_{w,A})$ and $h(c_{w,B})$ gives a hint as to losing bids. The solution of *Problem 1* satisfies robustness against “collusive attacks” as stated in Sect. 4.2. Therefore, we claim that the technique used in Selection of successful bid-2 gives an efficient solution to *Problem 1*.

A.1 Previous Work

Yao [Yao82] suggested a solution to the *millionaires problem* which can compare two party’s riches without leaking their values. This solution is closely related to *Problem 1*. In a more recent study, Jakobsson and Yung [JY96] presented a way of judging whether two values were equal or not without leaking them. Naor and Pinkas [NP99] solved a similar problem using the Oblivious Polynomial Evaluation (OPE), which was introduced in the same paper. Both of their approaches solve *Problem 1*, but they need many modular exponentiations. Here we briefly sketch their works in [NP99].

Oblivious Polynomial Evaluation

A polynomial P is known to Bob and he would like to let Alice compute the value $P(x)$ for input x known to her in such a way that neither Bob learns x , nor Alice gains any additional information about P .

OPE realizes various computations while keeping information secret; it solves *problem 1* as shown below:

Suppose that Alice has a secret value α_i and a polynomial P_A and Bob has α_j and P_B in the same way. They send $P_B(\alpha_i)$ and $P_A(\alpha_j)$ to each other using the OPE protocol. They then send $P_A(\alpha_i) + P_B(\alpha_i)$ and $P_A(\alpha_j) + P_B(\alpha_j)$ to each other.

They are convinced $\alpha_i = \alpha_j$ if and only if $P_A(\alpha_i) + P_B(\alpha_i) = P_A(\alpha_j) + P_B(\alpha_j)$ on the assumption where the probability $P_A(\alpha_i) + P_B(\alpha_i) = P_A(\alpha_j) + P_B(\alpha_j)$ for any $\alpha_i \neq \alpha_j$ is negligible.

³ If the number of elements in \mathcal{A} is sufficiently large, Problem 1 can be solved by checking the equality of $f(\alpha_i)$ and $f(\alpha_j)$ where f is, as one instance, a one-way hash function.

A.2 Our Solution

Problem 1 has been solved in previous works [JY96], [NP99], and their works can be applied our auction protocol as stated in Sect. 4. However, this section presents another approach. This is because their methods need many modular exponentiations to keep information secret. We present a faster protocol for *problem 1* assuming two auctioneers perform the protocol honestly and consider this assumption is reasonable by *verifiability* discussed in Sect. 5.

Our solution to *problem 1* needs only four modular exponentiations, as follows:

Let G be a finite cyclic group of order p where p is a large prime, and let $\alpha_0, \alpha_1, \alpha_2 \in_R G$ be given.

1. Alice chooses $d \in_R \mathbb{Z}/p\mathbb{Z}$, computes α_i^d and sends it to Bob.
2. Bob chooses $e \in_R \mathbb{Z}/p\mathbb{Z}$, computes α_j^e and sends it to Alice.
3. Alice computes $(\alpha_j^e)^d$ and sends it to Bob.
4. Bob computes $(\alpha_i^d)^e$ and sends it to Alice.

They are convinced $i = j$ if and only if $(\alpha_j^e)^d = (\alpha_i^d)^e$ with overwhelming probability.

The security of the above statement can be reduced to the modified DDH assumption under the condition that they are honest. To explain the assumption, a brief definition of the DDH assumption is given below (see [Bon98], [NR97] for further details).

Definition 1 ((DDH assumption)). Let G be a finite cyclic group of order p where p is a large prime, and let g be a generator of G . Then,

$$\{g^a, g^b, g^{ab} : a, b \in_R \mathbb{Z}/p\mathbb{Z}\}, \{g^a, g^b, g^c : a, b, c \in_R \mathbb{Z}/p\mathbb{Z}\}$$

are computationally indistinguishable.

You know that the order of all elements in G is prime p , except the unit. From this point, we consider that there is no advantage in how to choose g in *Definition 1* and if this conjecture is true, the below *modified DDH assumption* is implied under DDH assumption.

Definition 2 ((modified DDH assumption)). Let G be a finite cyclic group of order p where p is a large prime. Then, for any $g \in G$,

$$\{g^a, g^b, g^{ab} : a, b \in_R \mathbb{Z}/p\mathbb{Z}\}, \{g^a, g^b, g^c : a, b, c \in_R \mathbb{Z}/p\mathbb{Z}\}$$

are computationally indistinguishable.

Of course, g^{ab} and g^c are indistinguishable in the special case where g is a unit.

Next, using the fact that the matching Diffie-Hellman problem can be reduced to the DDH assumption (see [Bon98], [NR97]), we can reach the below statement.

Lemma 1. *The following problem can be reduced to the modified DDH assumption.*

Let G be a finite cyclic group of order p where p is a large prime. Then, find the value $\sigma \in \{0, 1\}$ such that

$$\{g^d, g^{x_0}, g^{x_1}, g^{dx_\sigma} : d, x_0, x_1 \in_R \mathbb{Z}/p\mathbb{Z}\}$$

where g is a generator of G and a distinguisher can choose it arbitrarily.

We omit the proof.

Finally, we need the below theorem to achieve our efficient auctions protocol in Sect. 4

Theorem 1. *Under the modified DDH assumption, Alice and Bob know only whether they choose the same value in problem 1 when they perform problem 1 honestly.*

Proof. The case of $i = j$ is trivial. For the case of $i \neq j$, suppose that Bob chooses α_j and would like to know Alice's secret value α_i ($i = j \pm 1 \pmod{3}$). Without loss of generality, you can replace α_j^e with $g \in G$, $\alpha_{j-1 \bmod 3}$ with g^{x_0} , and $\alpha_{j+1 \bmod 3}$ with g^{x_1} . If Bob finds Alice's secret value $\alpha_i = g^{x_i}$ by receiving $\alpha_i^d = g^{dx_i}$ and $\alpha_j^{ed} = g^d$, he would find σ in Lemma 1. \square

Oblivious Image Watermarking Robust against Scaling and Geometric Distortions^{*}

Francesc Sebé and Josep Domingo-Ferrer

Universitat Rovira i Virgili, Dept. of Computer Engineering and Mathematics, Av.
Països Catalans 26, E-43007 Tarragona, Catalonia, Spain
e-mail {fsebe,jdomingo}@etse.urv.es

Abstract. Watermarking stays the main technical safeguard of electronic copyright. This paper presents the first public-domain oblivious watermarking scheme for images which survives scaling and geometric distortion attacks. Previous proposals are either proprietary, non-oblivious or require scaling or geometric distortion to be undone prior to mark recovery, which may not be practical in oblivious watermarking. The new scheme uses a tile-based embedding technique that allows mark recovery from a scaled or geometrically distorted watermarked image. Other properties of the scheme presented here are that it operates on the spatial domain, supports multiple marking and does not require previous knowledge of the embedded copyright sequence. The latter property makes the proposed watermarking suitable for fingerprinting purposes.

Keywords: Oblivious watermarking, Multimedia copyright protection.

1 Introduction

The failure of sophisticated copy prevention systems like DVD [12] leaves copy detection, and more specifically watermarking, as the main solution for protecting the copyright of images or information in electronic format. In watermarking, the merchant embeds a mark in the copy sold and can later recover the mark from a redistributed copy to prove ownership or identify the redistributing buyer. Published watermarking schemes can be divided into oblivious and non-oblivious; the first class is more convenient for large-scale mass protection of digital content because it does not require the original content for mark recovery, while the second class does.

Oblivious watermarking offers a greater organizational flexibility and is better adapted to distributed copy detection than non-oblivious watermarking. For example, it enables the merchant to delegate copy detection to a set of agents distributed over the Internet, who can recover marks from intercepted redistributed content without having been entrusted with the original content. Such an arrangement minimizes disclosure of the original unprotected content (which

^{*} This work is partly funded by the Spanish CICYT under contract no. TEL98-0699-C02-02.

stays only known to the merchant) and also minimizes storage requirements (agents do not have to store the original version of all digital content they can come across of).

Commercial oblivious watermarking schemes surviving a broad range of manipulations (*e.g.* Digimarc [4]) tend to be based on proprietary algorithms not available in the literature.

1.1 Plan of This Paper

Section 2 is an overview of the literature on oblivious watermarking. Section 3 highlights the most outstanding features of our contribution in comparison to proposals in the literature. In Section 4, our procedure for mark embedding is described. Section 5 deals with the procedure for mark recovery. Parameter choice is discussed in Section 6. Empirical results on imperceptibility are given in Section 7. Empirical results on robustness are given in Section 8. Conclusions and topics for future research are summarized in Section 9.

2 Background on Oblivious Watermarking

There are two shortcomings affecting oblivious watermarking systems in the literature:

- Many published proposals require the embedded sequence to be given as an input to the mark detection procedure.
- No proposal in the literature embeds marks so that they can survive scaling and/or geometric distortion attacks.

In the two subsections below, we explain the two shortcomings above in more detail.

2.1 Systems That Require Knowledge of the Embedded Sequence

Many recently published oblivious schemes require previous knowledge of the embedded sequence before mark detection. This requirement makes mark recovery more robust but less flexible as the merchant needs to know beforehand which sequence she is looking for. Assuming such knowledge is definitely unrealistic if watermarking is used for fingerprinting (where the merchant embeds a different serial number or buyer ID in each copy being sold [10, 15]).

Examples of schemes with this problem are [10, 2, 7, 8]. In these proposals, the watermark takes the form of a Gaussian or binary pseudo random sequence s which is embedded in some transform domain. Let $C = \mathcal{T}(I)$, where \mathcal{T} denotes some transform and C are the transform coefficients of the original unmodified image I . A subset c of C is modified to embed the watermark. Let $\hat{C} = c \cup \bar{c}$, where $c \cap \bar{c} = \emptyset$. Denoting by \mathcal{E} the watermark embedding function, the overall embedding operation can be expressed as

$$C = \mathcal{T}(I) \quad \hat{c} = \mathcal{E}(c, s) \quad \hat{C} = \hat{c} \cup \bar{c} \quad \hat{I} = \mathcal{T}^{-1}(\hat{C})$$

Let $\tilde{I} = \hat{I} + N$ be the image in which the presence of the watermark is tested, where N is some noise that can appear between mark embedding and mark detection. The detection operation can be expressed as

$$\tilde{C} = \mathcal{T}(\tilde{I}) \quad \tilde{s} = \mathcal{D}(\tilde{c}) \quad s_d = \frac{s^T \tilde{s}}{|s| |\tilde{s}|}$$

where \mathcal{D} is a detector function and s_d is the detection statistic ($-1 \leq s_d \leq 1$) which is a measure of the normalized correlation of the embedded and detected signature sequences. In these schemes, an image \tilde{I} is considered to contain the watermark s if s_d is greater than a fixed threshold. Of course, computing s_d requires previous knowledge of s .

2.2 Scaling and Geometric Distortion Attacks

To our best knowledge, robustness against scaling and geometric distortion attacks is not achieved by any published oblivious scheme. Some previous schemes assume that such attacks can be undone prior to mark recovery [10, 7]. Undoing them requires knowledge of the original image which turns those schemes into non-oblivious ones.

In [1] an iterative search technique is used to cope with geometric attacks. The search technique seeks to emulate the inverse operation of the attack. It consists of running the mark recovery algorithm after trying various inverse operations until the bit error rate of the hidden bits drops dramatically or a high correlation with the original watermark is obtained. Thus, even if this system does not generally require knowledge of the embedded mark for recovery, it definitely requires such knowledge in order to survive geometric attacks. Furthermore, the search technique used to undo such attacks is too expensive and cannot be applied to random distortion attacks where the inverse operation is unknown.

In [3], image pixels are permuted using a pseudo random seeded permutation. Then the permuted image is divided into blocks and information is embedded by modifying the standard deviation of gray (or color) levels of the pixels in each block. Finally, the permutation is undone. During mark recovery, the image is again permuted and the embedded watermark is extracted. The problem is that, if the watermarked image has undergone geometric distortion, permuting it and dividing it into blocks will yield a block partition completely different from the one used in the embedding process, so that the recovered mark will be incorrect.

In other oblivious schemes, geometric attacks are left for future research [2, 8] or are not even mentioned [6, 3]. It is actually unclear whether the latter two proposals can be termed robust, because they are very sensitive to geometric distortion.

We next sketch the fundamentals of the widely used method [6], so that its vulnerability to geometric distortion attacks can be better understood. This method uses a spread spectrum technique to obtain an oblivious watermarking system. The embedding and recovery procedures are as follows:

Embedding. The copyright information to be embedded is a binary sequence a_j , $a_j \in \{-1, 1\}$. This discrete signal is spread by a large factor cr , called chip-rate, to obtain the sequence $b_i = a_j$, $j \cdot cr \leq i < (j+1) \cdot cr$. The spread sequence b_i is amplified by a locally adjustable amplitude factor $\alpha_i \geq 0$ and is then modulated by a binary pseudo-noise sequence p_i , $p_i \in \{-1, 1\}$. If v_i is the original signal to be protected, the resulting watermarked signal is $\tilde{v}_i = v_i + \alpha_i \cdot b_i \cdot p_i$.

Recovery. Mark recovery is performed by demodulating the watermarked signal with the same pseudo-noise signal p_i that was used for embedding, followed by summation over the window for each embedded bit, which yields the correlation sum s_j for the j 'th information bit $s_j = \sum_{i=j \cdot cr}^{(j+1) \cdot cr - 1} p_i \cdot \tilde{v}_i$. The sign of s_j is interpreted as the embedded bit a_j . To generate a correct result, each pixel \tilde{v}_i must be demodulated by the corresponding p_i of the random sequence. This requirement turns this scheme very vulnerable to geometric distortion attacks which can cause the synchronization between sequences \tilde{v}_i and p_i to be lost.

3 Our Contribution

In addition to being more robust, the oblivious watermarking scheme presented in this paper is simpler than the aforementioned ones in that it operates in the spatial domain rather than in transformed domains (DCT, wavelet). It does not require prior knowledge of the embedded sequence and survives all kinds of scaling and moderate geometric distortion attacks (*e.g.* row and column removal, shearing, random distortion and small cropping and rotation) without requiring the original image for mark recovery. It is also robust against most filtering and compression attacks performed by the StirMark 3.1 benchmark (see [9] and below).

4 Mark Embedding

Without loss of generality, we will assume a monochrome image in what follows; for RGB color images, watermarking is independently done for each color plane. Let the original image be $X = \{x_i : 1 \leq i \leq n\}$, where x_i is the color level of the i -th pixel and n is the number of pixels in the image. Let x_i take integer values between 0 and $MAXCOLOR$, so that the lower x_i , the darker is the color level. Let $dt \in [0, MAXCOLOR]$ be a threshold such that all color levels x_i below dt visually appear as dark.

An algorithm is next given whereby the merchant can compute pixel visual components, that is, the perceptual value of pixels. This value is an estimate of the maximum subperceptual increment/decrement that each pixel can accommodate. The idea underlying Algorithm 1 is that dark pixels and those pixels in non-homogeneous regions are the ones that can best accommodate embedded information while minimizing the perceptual impact. In Algorithm 2 below, let

lb_1 and ub_1 be integer values $lb_1 < ub_1$ that are used as parameters to bound the variation of pixel color values. For a given $MAXCOLOR$, suitable values for dt , lb_1 and ub_1 are empirically chosen; for example, for $MAXCOLOR = 255$ (8-bit color level), a good choice is $dt = 70$, $lb_1 = 2$, $ub_1 = 11$.

Algorithm 1 (Visual components(dt, lb_1, ub_1))

1. For $i = 1$ to n do:
 - a) Compute $m_i := \max_j |x_i - x_j|/2$, for all pixels j which are neighbors of pixel i on the image (there are up to eight neighbors); m_i can be regarded as a kind of discrete derivative at pixel i . To bound the value of m_i between lb_1 and ub_1 , perform the following corrections:
 - i. If $m_i > ub_1$ then $m_i := ub_1$.
 - ii. If $m_i < lb_1$ then $m_i := lb_1$.
 - b) Compute the darkness of the i -th pixel as $d_i := (dt - x_i) * ub_1 / dt$ if $x_i < dt$ and $d_i := 0$ otherwise. A pixel is considered as dark if its color level is below dt . The value of d_i lies between 0 and ub_1 .
 - c) Compute the preliminary visual component of the i -th pixel as $v_i := \max(m_i, d_i)$.
2. For $i = 1$ to n compute the final visual component of the i -th pixel as $V_i := \max_j v_j$, for all pixels j which are neighbors of i on the image plus the pixel i itself.

The higher V_i for a pixel, the less perceptible are changes in that pixel. The merchant is now ready to embed a copyright binary sequence in the image. The embedding process is governed by a key k known only to the merchant and is based on incrementing/decrementing the pixel color level. The absolute variation for a pixel is always less than its visual component. Integers lb_2 and ub_2 in the algorithm below are parameters such that $lb_2 < ub_2$. Given $MAXCOLOR$, a suitable value is determined for these parameters (see also Section 6). For example, $lb_2 = 10$ and $ub_2 = 13$ are good choices for $MAXCOLOR = 255$.

Algorithm 2 (Mark embedding(k, lb_2, ub_2))

1. If ε is the binary sequence to be embedded, encode ε using an error-correcting code (ECC) to obtain the encoded mark E .
2. Using the key k as seed, pseudo-randomly place $|E|$ non-overlapped square tiles R_i over the image, where $|E|$ is the bitlength of E . Tile size is also determined by k .
3. Using k as seed, pseudo-randomly assign a value a_i between lb_2 and ub_2 to tile R_i , for $i = 1$ to $|E|$.
4. To embed the i -th bit e_i of the mark E in R_i :
 - a) Divide the color level interval $[0, MAXCOLOR]$ into subintervals of size a_i .
 - b) Label consecutive subintervals alternately as “0” or “1”.

- c) For each pixel x_j in R_i :
 - i. If x_j lies in a subinterval labeled e_i , bring it as close as possible to the interval center by increasing or decreasing x_j no more than V_j .
 - ii. If x_j lies in a subinterval labeled \bar{e}_i , bring it as close as possible to the nearest neighbor interval center (neighbor intervals are labeled e_i) by increasing or decreasing x_j no more than V_j .

5 Mark Recovery

Upon detecting a redistributed image \hat{X} , \hat{e} can be recovered as follows, provided that the length $|E|$ of the embedded mark and the secret key k used for embedding are known (the merchant should know these parameters).

Algorithm 3 (Mark recovery(k, lb_2, ub_2))

1. Using the key k as seed, pseudo-randomly place $|E|$ non-overlapped square tiles R_i over the image (again, tile size is also determined by k). Also using k as seed, pseudo-randomly assign a value a_i between lb_2 and ub_2 to tile R_i , for $i = 1$ to $|E|$. (Tiling done in this step is analogous to tiling done during mark embedding).
2. To recover the i -th bit \hat{e}_i of \hat{E} from R_i :
 - a) Divide the color level interval $[0, MAXCOLOR]$ into subintervals of size a_i .
 - b) Label consecutive subintervals alternately as “0” or “1”.
 - c) Let $ones := 0$ and $zeroes := 0$.
 - d) For each pixel x_j in R_i :
 - i. If x_j lies in a subinterval labeled “1”, then $ones := ones + 1$
 - ii. If x_j lies in a subinterval labeled “0”, then $zeroes := zeroes + 1$
 - e) If $ones > zeroes$ then $\hat{e}_i := 1$; if $ones < zeroes$ then $\hat{e}_i := 0$; otherwise \hat{e}_i is an erasure.
3. Decode \hat{E} with the same ECC used for embedding to obtain \hat{e} .

6 Parameter Choice

In Algorithm 1 (visual components), suitable values for parameters dt , lb_1 and ub_1 should be empirically chosen for a given $MAXCOLOR$. Suitability depends on visual perception and robustness considerations. We have suggested above a good choice for $MAXCOLOR = 255$, namely $dt = 70$, $lb_1 = 2$ and $ub_1 = 11$; for other values of $MAXCOLOR$, a rule of thumb of is to scale that choice by $MAXCOLOR/255$. We discuss below other parameters related to Algorithm 2 (embedding) and Algorithm 3 (recovery).

6.1 On the Size of Tiles

At Step 2 of the mark embedding algorithm, $|E|$ square tiles are randomly placed over the image. From the point of view of robustness, the tile size must be large enough so that each bit is embedded in a sufficient number of pixels. However, the requirement that all $|E|$ tiles should be placed non-overlappingly limits the maximum tile size, which decreases as $|E|$ increases.

An additional consideration is imperceptibility. Better imperceptibility is gained if neighboring tiles are separated by a band of unmodified pixels. This further limits the tile size.

6.2 On the Width of Color Level Subintervals

The size a_i in which we divide the color level interval is a tradeoff between robustness and imperceptibility:

- Making such intervals narrow means that, during the mark embedding algorithm, the variation applied to pixels to mark them is low, which leads to better imperceptibility. The drawback of narrow intervals is a loss of robustness, because even small noise can easily shift a color level to a neighboring subinterval, which can cause an incorrect bit to be recovered.
- Larger values a_i yield higher robustness, but moving the color level of a pixel to a neighboring subinterval can be perceptible.

Thus, given $MAXCOLOR$, the interval $[lb_2, ub_2]$ where a_i randomly takes values has its lower bound lb_2 limited by robustness and its upper bound ub_2 limited by imperceptibility.

7 Imperceptibility Assessment

The scheme was implemented with parameter values $MAXCOLOR = 255$, $dt = 70$, $lb_1 = 2$, $ub_1 = 11$, $lb_2 = 10$, $ub_2 = 13$. The error-correcting code used was a dual binary Hamming code $DH(31, 5)$. If there 7 or less errors in a codeword, this code guarantees correction; for more than 7 errors, correction is not guaranteed.

The following images from [13] were tried: Lena, Bear, Baboon and Peppers. A 30-bit long mark ε was used, which needed six codewords of the dual Hamming code and resulted in an encoded E with $|E| = 31 \times 6 = 186$ bits. For Lena, a version of size 512×512 pixels was used and the length of the tile side was randomly chosen between 11 and 31; for the other images, a similar proportion between image size and tile size was maintained. For all images, tiles were placed so that neighboring tiles were separated by a band of unmodified pixels at least one pixel wide.

Figure 1 shows the original and the marked Lena; the Peak Signal-to-Noise Ratio (PSNR) between both images is as high as 41.16 dB.



Fig. 1. Left, original Lena. Right, Lena after embedding a 30 bit length mark (PSNR=41.16 dB)

7.1 Multiple Marking

A useful feature of the presented algorithm is that multiple marking is supported. Up to three consecutive markings on the same image are possible without substantial perceptual degradation nor loss of robustness. For example, the content creator M_1 can mark an image and sell the marked image to a distributing company M_2 which re-marks the image with its own mark, re-sells it to a retailer M_3 , who re-marks the image again before selling it to the end consumer. Each of M_1 , M_2 , M_3 can recover their embedded watermark by using the same key they used at embedding time.

Each of the four test images was marked three times and the results shown in Table 1 were obtained. The table shows the PSNR between each original image and the successive marked versions of it; the more markings, the lower is PSNR, which means more degradation of perceptual quality. The table also shows the average number of errors corrected per codeword of the encoded copyright sequence during mark recovery; this average is computed over the six codewords used to encode the mark. The number of errors per codeword increases with the number of consecutive markings and puts a strict limitation on how many times the image can be re-marked: if a codeword recovered from the marked image contains more errors than the correcting capacity of the ECC being used (7 errors in our implementation), then the mark might not be correctly recovered.

To illustrate the effects on imperceptibility, Figure 2 shows Lena after three consecutive markings.

8 Robustness Assessment

Some general considerations regarding robustness in front of the various kinds of attacks are as follows:

Table 1. Variation of perceptual quality and error rate for consecutive markings

	1st marking		2nd marking		3rd marking	
Image	PSNR (dB)	Errors	PSNR (dB)	Errors	PSNR (dB)	Errors
Lena	41.2	1	38.2	1.16	36.4	2.6
Peppers	39.4	0.16	36.2	1.83	34.5	2.6
Bear	39	0.16	35.4	1.33	33.7	3.1
Baboon	37.5	0	34.2	0.8	32.4	4



Fig. 2. Lena after three consecutive markings

- After an attack, a bit is correctly recovered if a majority of correct mark bits are still inside the corresponding tile.
- Scaling attacks are survived by placing tiles in positions relative to the image size. In this way, even if the image size varies, each tile still contains original mark bits.
- Unless tiles are very small, other attacks like row and column removal, shearing, cropping and rotation will only succeed if most pixels in the tile suffer a variation so large that it leads to visual degradation.

The base test of the StirMark 3.1 benchmark [9] was used to evaluate robustness on the marked versions of the four test images. The same parameter values listed at the beginning of Section 7 were taken and the following manipulations were survived:

1. Color quantization
2. Most low-pass filtering manipulations. More specifically:
 - a) Gaussian filter (blur)
 - b) Median filter (2×2 , 3×3 and 4×4).
 - c) Linear filter
3. JPEG compression for qualities 90 down to 30.

4. All StirMark scaling attacks (scale factors from 0.5 to 2).
5. All StirMark aspect ratio modification attacks.
6. All StirMark row and column removal attacks.
7. All StirMark shearing attacks.
8. Small rotations with and without scaling from -2 to 2 degrees.
9. Small cropping up to 2%.
10. StirMark random bend.

Robustness against different attacks has been measured taking into account the average number of errors corrected per codeword of the encoded copyright sequence. Table 2 shows the average number of errors per codeword when recovering the mark from the watermarked image Lena after an attack with low-pass filters and color quantization. The proposed scheme is very robust against this kind of attacks, since the average number of errors is well below the error-correcting capacity of the ECC used (7 errors).

Table 2. Average number of errors per codeword in mark recovery after low-pass filtering and color quantization (Lena)

Attack	Errors/codeword
Gaussian	3.16
Median (2×2)	1
Median (3×3)	1.5
Median (4×4)	1.83
Linear	2.83
Color quantization	2

Table 3 shows the average number of errors per codeword after compressing the watermarked image Lena with different JPEG quality levels. It can be seen that the average number of errors grows linearly with the compression rate (inverse of JPEG quality level). Figure 3 shows the watermarked Peppers image after a compression attack at JPEG quality 30; the mark is still recoverable.

Table 4 shows the average number of errors per codeword when recovering the mark from the watermarked image Lena after scaling attacks. It can be seen that very few errors are introduced when the scaling factor is greater than 0.90. The number of errors increases linearly as the scaling factor decreases toward 0.50. For Lena, recovery is still correct after 0.50 scaling in spite of there being an average 5.83 errors/codeword (see Figure 4). If there was some codeword with more than 7 errors, recovery could fail; this does not happen for any of the images tried, but could happen for other images after a 0.50 scaling attack.

Another family of StirMark attacks are aspect ratio modifications in the x and y directions with scaling factors between 0.8 and 1.2. The average number of errors introduced by such attacks is rather low (about 1.5 errors/codeword). Thus, the damage inflicted by these attacks is similar to the damage caused by moderate scaling attacks.

Table 3. Average number of errors per codeword in mark recovery after JPEG compression (Lena)

JPEG quality	Errors/codeword
90	1.16
80	1.83
70	2.16
60	2.66
50	3.66
40	4.83
30	5.16



Fig. 3. Marked Peppers compressed at JPEG quality 30 (mark still recoverable)

Table 4. Average number of errors per codeword in mark recovery after scaling (Lena)

Scaling factor	Errors/codeword
0.50	5.83
0.75	2.5
0.90	1.33
1.10	1.33
1.50	1.33
2.00	0.5

Table 5 shows robustness results after rotation with and without scaling on the watermarked Lena. In rotation without scaling, the rotated image is cropped so that it completely fills a horizontal frame (the resulting image is smaller than the original); in rotation with scaling, the rotated image is cropped and then magnified so that the horizontal frame being filled is of the same size as the original image. For small rotations, the average number of errors per codeword

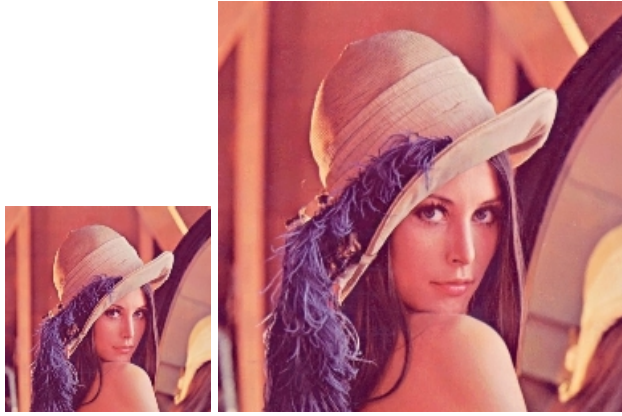


Fig. 4. Left, Lena after a 50% scaling attack. Right, Lena after the StirMark random bend attack. The mark is correctly recovered from both images.

is similar with and without scaling; for each rotation angle, Table 5 gives the joint average of the average number of errors per codeword in both cases.

Table 5. Average number of errors per codeword in mark recovery after rotation with and without scaling (Lena)

Rotation (degrees)	Errors/codeword
0.25	1.65
0.50	1.87
0.75	2.54
1.00	2.91
2.00	3.45

Small croppings and the StirMark random bend are also survived for the four images tried. For Lena, 1% cropping is survived with an average of 1.66 errors per codeword and 2% cropping with 2.16 errors. Also for Lena, the StirMark random bend attack is survived with an average of 3.33 errors per codeword (see Figure 4).

9 Conclusions and Future Research

Oblivious watermarking is attractive because it allows the content owners to delegate redistributor tracing to agents that need not be entrusted with the original content. However, oblivious watermarking methods in the literature fail to survive geometric distortions. We have presented here the first oblivious system which offers robustness against a comprehensive range of such attacks. In

addition, the proposed system operates in the spatial domain and is thus conceptually simpler than most previous oblivious systems.

Future research will be devoted to further enhancement of the robustness of oblivious watermarking systems. A desirable achievement would be to add all StirMark croppings to the list of survived attacks.

Another topic that should be investigated both for oblivious and non-oblivious watermarking is tamper-proofness. Benchmarks like StirMark are useful to evaluate robustness against standard signal processing attacks, but they do not deal with tamper-proofness, *i.e.* with attacks that exploit knowledge of the algorithm internal operation. If watermarking is to conform to Kerchhoff's assumption of algorithms being public-domain, tamper-proofness becomes a relevant issue.

References

1. F. Alturki and R. Mersereau, "Robust oblivious digital watermarking using image transform phase modulation", in *International Conference on Image Processing - ICIP'2000*, IEEE Signal Processing Society, 2000.
2. M. Caramma, R. Lancini, F. Mapelli and S. Tubaro, "A blind & readable watermarking scheme for color images", in *International Conference on Image Processing - ICIP'2000*, IEEE Signal Processing Society, 2000.
3. P.-M. Chen, "A robust digital watermarking based on a statistical approach", in *International Conference on Information Technology: Coding and Computing - ITCC'2000*, IEEE Computer Society, 2000, pp. 116-121.
4. Digimarc, <http://www.digimarc.com>
5. J. Domingo-Ferrer and J. Herrera-Joancomartí, "Short collusion-secure fingerprints based on dual binary Hamming codes" *Electronics Letters*, 36(20): 1697-1699, Sep. 2000.
6. F. Hartung and B. Girod, "Watermarking of uncompressed and compressed video", *Signal Processing*, 66(3): 283-301, May 1998.
7. C.-S. Lu and H.-Y. Mark Liao, "Oblivious cocktail watermarking by sparse code shrinkage: a regional- and global-based scheme", in *International Conference on Image Processing - ICIP'2000*, IEEE Signal Processing Society, 2000.
8. C.-S. Lu and H.-Y. Mark Liao, "An oblivious and robust watermarking scheme using communications-with-side-information mechanism", in *International Conference on Information Technology: Coding and Computing - ITCC'2001*, IEEE Computer Society, 2001, pp. 103-107.
9. F. A. P. Petitcolas, R. J. Anderson and M. G. Kuhn, *StirMark v3.1* <http://www.cl.cam.ac.uk/~fapp2/watermarking/stirmark/>
10. M. Ramkumar and A. N. Akansu, "A robust oblivious watermarking scheme", in *International Conference on Image Processing - ICIP'2000*, IEEE Signal Processing Society, 2000.
11. N. R. Wagner, "Fingerprinting", in *1983 IEEE Symposium on Security and Privacy*, Oakland CA: IEEE, 1983, pp. 18-22.
12. <http://www.lemuria.org/DeCSS>
13. http://www.cl.cam.ac.uk/~fapp2/watermarking/benchmark/image_database.html

Fingerprinting Text in Logical Markup Languages

Christian D. Jensen

Department of Computer Science
Trinity College Dublin,
Dublin 2, Ireland
`Christian.Jensen@cs.tcd.ie`

Abstract. Information hiding is attracting an increasing attention from the research community. Most of this research has centered around hiding information, such as watermarks and fingerprints, in images or digital audio and video signals. Text has generally been treated as a black & white image with special properties. All of the current methods of hiding information in text are vulnerable to scanning followed by optical character recognition in order to reconstruct the text.

Document distribution is increasingly relying on logical markup languages like HTML and XML, where the physical presentation of the text is determined by the user's browser. Embedding the watermark in the physical presentation of the document is therefore no longer practical. We argue that embedding syntactic or semantic fingerprints in text is the only viable way to fingerprint document in logical markup languages such as HTML or XML.

In this paper, we propose a new semantic fingerprinting mechanism based on synonym substitution. This idea is developed into an operational system and results of preliminary experiments are reported.

1 Introduction

Information hiding is attracting increasing attention from the research community [19]. The driving force behind much of this effort is the wish to embed fingerprints in all forms of digital media distributed over the Internet. Most of this research has centered on sound, image and multimedia distribution, but a considerable effort has been put into watermarking text and detection of copyright violations [12,4,5,14,15,16,17,18,21,22,23].

Digitally stored text documents are easy to copy and redistribute. The ability to dissuade copyright violations through embedding hidden watermarks and fingerprints in digital documents is therefore often considered an enabling technology for electronic publishing and digital libraries [18,23]. Watermarks are associated with authentication of a document's source (the copyright holder), while a digital fingerprint is a distinct watermark, such as a serial number, embedded in every copy of a document. Thus the fingerprint allows the copyright owner to trace the origin of illegal copies and prosecute the copyright violator. The ability to trace the source of all copies is useful in identifying the source of leaked documents, which is also known as traitor tracing.

Most of the existing methods for watermarking and fingerprinting text rely on the physical markup of the document, either changing the line spacing, the inter word spacing

or slightly modifying the font. These techniques are all vulnerable to scanning followed by optical character recognition (OCR) in order to reconstruct the text [16]. Moreover, they are completely unsuitable for logical document markup languages such as HTML [20] or XML [3], where the final rendering of the document is controlled by a program (browser) running on the user's machine.

Logical markup already plays an important role in Electronic Data Interchange (EDI) and on the World Wide Web. The prevalence of web browsers means that computer manuals and computer magazines are increasingly delivered as a hierarchy of web-pages in HTML on a CD-ROM. This development is likely to spread to other areas of document distribution, such as electronic books and journals and access to digital libraries.

The techniques of embedding syntactic and semantic watermarks in text are largely ignored by the research community. We argue that embedding syntactic and semantic fingerprints in the document is the only viable way to fingerprint documents in logical markup. In this paper we propose a new semantic fingerprinting mechanism based on synonym substitution. The idea of embedding a watermark by substitution of synonyms is proposed elsewhere [1], but they do not develop the idea into an operational system.

The synonym substitution technique presented in this paper is to our knowledge the first description of an operational semantic fingerprinting system. The technique is based on extracting features from the text, associating users with distinct subsets of feature values and textual transformations that preserve the semantics of the document, while matching the extracted features to the set of values associated with the intended recipient of the document.

We evaluate the proposed mechanism in the context of traitor tracing, where a distinct fingerprint is embedded in every copy of an electronically distributed document. Traitor tracing has unique properties that makes it simpler than fingerprinting for detection of copyright violators. First, the system only needs to recognize one fingerprint per user, i.e., the same fingerprint can be embedded in every document distributed to the same user. Second, the system only needs to cater for a relatively small number of authenticated users, because only a few copies of confidential documents are normally circulated. Finally, traitors are normally acting alone, so collusion among traitors is highly unlikely.

The rest of this paper is structured in the following way: Section 2 describes our document model and present related work on watermarking and fingerprinting text. Our semantic watermarking mechanism is presented in Section 3. The mechanism is evaluated in Section 4 and conclusions and future work are presented in Section 5.

2 Fingerprinting Digital Documents

In order to facilitate the discussion of related work, we start by introducing our document model.

2.1 Document Model

A document consists of three main components: the *message* (the semantic content encoded in the document), the *markup* (the structure and physical layout of the document)

and the *presentation* (the bitmap presented to the user on paper or on a computer screen). A text processing system is used to combine the content and the markup in order to produce the physical presentation of the document. This process is called rendering the document.

Document Structure. A document is considered an *ordered hierarchy of content objects* (OHCO) [9], e.g., a scientific paper consists of a title, an author, an affiliation, an abstract, a number of sections and a bibliography. Each of the sections may in turn be composed by a sequence of nested subsections. At the lowest level, subsections are composed from *basic text units*, such as sentences, paragraphs, quotations, equations, theorems, proofs, etc. Many of these basic text units can be decomposed further, e.g., sentences consist of words and words consist of syllables. This hierarchical model corresponds well to the structure of documents in most logical markup languages, e.g., XML.

Markup. The markup describes the logical structure of the document (logical markup) and the physical lay-out of the document (physical markup). Both content and markup are present in the *source file* stored by the system. This source file is rendered by the user's document processing system before it is presented to the user.

The logical markup only describes the logical structure of the document, the physical appearance is determined by the user's document processing system at runtime, i.e., when the document is displayed on the screen or printed by the user.¹ The logical markup explicitly identifies the hierarchy of content objects, i.e., it breaks the document into smaller units, such as chapters, sections, subsections, paragraphs, sentences, phrases, etc. The larger units are normally identified by special tags (headers) while phrases and sentences normally are delimited by punctuation marks.

The physical markup describes how the document should be rendered by the user's document processing system. It defines how the document must appear in terms of indentation, fonts, inter-word and inter-line spacing, etc.

2.2 Related Work

There exist two ways of fingerprinting a document. Characteristic document features can be extracted from an existing document. These features will uniquely identify the document in the same way as a human fingerprint. The other possibility is to embed a fingerprint in the document. This fingerprint can later be extracted in order to identify a particular instance of the document.

Extracted Fingerprints. A number of projects have addressed the problem of extracting characteristic features from a document, to facilitate the identification of complete or partial copies of the document [4][5][14][23][22].

The fingerprint of a given document can be compared to a set of fingerprints stored in a database in order to identify cases of copyright violations or plagiarism.

¹ Logical markup is sometimes called descriptive markup [8].

An extracted fingerprint can be used to detect major similarities between two documents, but it is unable to detect small variations between two similar documents. It can therefore be used to detect a copyright violation, but it cannot be used to identify the perpetrator.

In this paper, we focus on embedded fingerprints, where the source file is modified in order to hide a serial number or the identity of the intended recipient in each copy of the document.

Embedded Fingerprints. In general, there exist three different ways to hide information, such as a fingerprint, in the source file of a given text: open space methods, syntactic methods and semantic methods [1]. A fourth method consists of embedding the fingerprint in the document bitmap (the rendered version of the document).

Open Space Fingerprints. An open space fingerprint can be embedded in both logical and physical markup. In logical markup, extra white space between words and tags can be used to encode a fingerprint. Such extra whitespace is normally ignored when the document is rendered, which makes the fingerprint invisible under normal circumstances. However, the fingerprint is visible in ordinary text editors and it is easy to write a program that eliminates all superfluous whitespaces, thus eliminating the fingerprint. Moreover, open space fingerprints are vulnerable to the OCR attack.

Syntactic Fingerprints. A fingerprint can be embedded in a document by manipulating the syntactic structures of the document. Many of the rules of punctuation are ambiguous or redundant, and mispunctuation can have little impact on the reader's understanding of the message. The serial comma is a good example of syntactic ambiguity: both phrases "Jack, Joe, and Jill" and "Jack, Joe and Jill" are considered correct usage. A fingerprint can be embedded in a text by alternating between the two types of serial comma, e.g., by attributing the value "1" to the first usage and "0" to the second usage. However, this inconsistent use of the serial comma is easy to notice and the fingerprint can be overwritten by changing all instances of the serial comma to a single form of usage.

The structure and form of sentences can also be used to embed a fingerprint in a text. The following two sentences convey the same message: "When I was a child, a lollipop only cost 5p" and "A lollipop only cost 5p when I was a child".

Syntactic fingerprints, may change the punctuation or the structure of the text, but the actual words must remain the same.

Semantic Fingerprints. A semantic fingerprint works by changing the actual words of the document, e.g., by substituting words with synonyms. Semantic fingerprinting appears to have been largely ignored by the scientific community and we have address this issue.

Raskin et Al. [21] have proposed a semantic watermarking technique based on determining whether the representation of selected words are quadratic residues or not. They outline an algorithm for inserting and detecting watermarks; this algorithm is both complex and computationally expensive. Moreover, fingerprint detection requires access to the original document, so at least one non-fingerprinted copy of the document must exist. Anyone with access to this original document can safely leak it. Finally, the outlined algorithms are very sketchy and no implementation is reported.

A German company, Compris.com, has developed two programs that employ semantic methods to either embed a watermark in a document (TextSign [12]) or hide a secret message in a document (TextHide [11]). The algorithm is not presented, but both programs rely on a large data base, possibly containing known words and their synonyms. The *same* database is used to embed and detect the fingerprint in the document. The technique presented in this paper only needs a database of synonyms to embed the fingerprint in the document. Moreover, this database can be changed at any given moment without affecting existing fingerprints.

The advantage of both syntactic and semantic fingerprints is that they resist the OCR attack.

Bitmap Fingerprints. A bitmap fingerprint is embedded in the presentation of the document, normally a bitmap in some form. The fingerprint can be encoded by subtle differences in inter-word and inter-line spacing, similar to the open space methods mentioned above. Bitmap fingerprints are proven to be robust with respect to copying [2, 18, 17, 15] and are well suited for printed documents. However, as with the open space fingerprints, bitmap fingerprints are vulnerable to the OCR attack.

3 Synonym Substitution

In order to fingerprint a document in logical markup, some transformation has to be applied to the actual text of the document without changing the semantics. The OHCO model guarantees that the semantic impact of a modification to a basic text unit is mostly confined to the encompassing branch of the hierarchy, and that the semantic impact of the modification decreases with the hierarchical distance to the modified text unit. This means that a semantically small transformation of a basic text unit is unlikely to significantly alter the overall meaning of the document. The transformation must uniquely identify the author or the intended recipient of the document, depending on the purpose of the fingerprint. Synonym substitution is one possible transformation which is described in the following, but many other semantics preserving transformations are possible.

3.1 Overview

A document is divided into its constituent basic text units, e.g., lines, sentences, paragraphs, sections or chapters. The choice of basic text units depends on the type and the size of the document and the number of required fingerprints. A function, which extracts the chosen features, is then applied to selected text units. We currently use a hash function, which generates a unique numerical value for that text unit. These hash values are the extracted features, which constitute the fingerprint of the document. If the document changes so do the hash values.

Each intended recipient must be assigned a *key* that uniquely identifies that user. The key defines a sequence of subsets that the extracted features must belong to, i.e., the key consists of a sequence of intervals that the hashed sentences must belong to. The key

is independent of the document and the same key can be used to mark all documents intended for the same recipient.

The document transformation must ensure that the fingerprint of the document matches the fingerprint of the intended recipient, e.g., by substituting one or more words from the original document with synonyms to change the hash values of the text unit.

3.2 Semantics Preserving Transformations

The applicability of semantic fingerprinting depends on the ability to identify text transformations that preserve the semantics of the document.

Some semantics preserving text transformations can be applied without considering the context, such transformations are considered *safe*, other transformations require either local context (the context of the text unit) or global context (the context of the document, e.g., the type of document and the subject matter) to be considered. Safe transformations can be performed automatically while context dependent transformations may require human intervention.²

Although some safe transformations exist, e.g., dative shifts, the opportunity to exploit them is limited, so we focus on one type of context dependent transformation: synonym substitution. Consequently, the transformations have to be assisted by human intervention in order to preserve the semantics of the document's message.

3.3 Hash Function Properties

Synonym substitution only affects a relatively small part of the text unit, this means that the function used to fingerprint the document must have the following property: a small change in the hashed string must result in a small change in the hash value.

$$H(T + \Delta) = H(T) + \delta$$

where Δ is a small change in the text domain and δ is a small numerical value.

A simple summation of the ASCII values of all the letters in the hashed text satisfy this requirement. However, the sum of ASCII values depend on the length of the text unit, i.e., longer text units have larger hash values. We therefore use the remainder of the sum modulo some value d . This introduces a discontinuity, but the condition is satisfied when $d < \delta$.

The value of d impacts the key in the following way. A large value of d can be used to widen the hash intervals of the key, thereby making it easier to transform the text to fit the key. Another possibility is to introduce more hash intervals, thereby shortening the length of the key sequence. Our preliminary experiments shows that a relatively small number works well, i.e., we used $d = 13$ in our experiment.

² The linguistic analysis required to automate context dependent transformations is beyond the scope of this paper.

3.4 Keys

The key consists of a sequence of legal intervals for a given user, e.g., $\text{user1} = [0 - 4], [6 - 9], [2 - 6]$ and $\text{user2} = [5 - 8], [4 - 8]$. Extracting the hash values 3, 7, 2 from the document identifies the source of the document as user1.

In addition to the sequence of intervals, a random number is associated with each user. This random number is used to seed a pseudo-random-number generator, which is used to spread the fingerprint over the entire text.

The key is used to determine both location and value of the fingerprint, so it must be kept secret from the users.

Key Length. The length of keys is variable, this means that new and longer keys can be added when needed. The length of the key impact the fingerprint in the following way. A long key means that a larger fraction of the document is used to carry the fingerprint. This increases the probability that an attacker can distort the fingerprint by random modifications, e.g., the same type of transformations that are used to fingerprint the document. On the other hand, a long key means that redundancy can be introduced into the key-space which reduces the probability of false positives.

Key Width. The hash function maps onto an interval, which defines the *overall width* of the key. This interval is divided into smaller *key intervals* that are assigned to the different users as part of their key. If the hash function maps onto the interval $[0 - 12]$, different users can be assigned the intervals $[0 - 4]$, $[6 - 10]$ and $[8 - 10]$. The key intervals need not be of the same size and the key intervals of different intervals may overlap for part of their keys. It is important that two keys do not overlap in all key intervals, because this could lead to ambiguous fingerprints.

3.5 Fingerprinting Documents

The fingerprint can be embedded by a document processing system every time a user saves a document. This way, all stored documents will contain the fingerprint of their creator (or the user who last edited the document). Communication software (web servers, email systems, file transfer programs) can also embed a fingerprint in a document before it is transferred to the intended recipient. This facilitates detection of the source of a leaked document.

Embedding the Fingerprint. The fingerprinting algorithm starts by selecting the first text unit to mark. The random value stored with the key is used to seed a pseudo-random-number generator, which selects the first position to mark. The number returned by the pseudo-random-number generator is added to the current position (initially 0) in order to select the first sentence to transform. This step is then repeated for every interval in the key. This means that the fingerprint of different users will be located at different positions in the document.

The marking algorithm calculates the hash of the text unit. If the hash value is outside the corresponding interval in the user's key, one of the words have to be substituted by

a synonym. The marking algorithm searches an online database of synonyms in order to identify one or more synonyms that will bring the hash value of the text unit inside the interval.³ The synonyms are then presented to the user, who chooses a synonym that preserves the message of the document; if no suitable synonym is found, the text unit has to be rephrased.

Detecting the Fingerprint. The fingerprint detector has to examine all users sequentially. The pseudo-random-number generator is seeded with the random number stored in the user’s key. The first sentence is hashed and compared to the first interval in the user’s key. This matching continues until there is a mismatch or the hashes match all the intervals. In the latter case, the fingerprint has been detected in the document.

4 Evaluation

An evaluation of fingerprinting with synonym substitution must determine to what extend a fingerprint can be embedded in a document without destroying the message. Our preliminary experiments show that this is possible and we present an example that illustrates this point.

4.1 Example of Synonym Substitution

We did not want to leak confidential information in this paper, so instead we have applied synonym substitution to a section from an instruction manual for a clock radio. The example uses sentences as the text unit, where anything between two full stops is interpreted as a sentence. The hash function calculates the sum of ASCII values of all the characters between the two full stops (excluding whitespaces) modulo 13. The sentences and the hash values of the original document is shown in Figure 1.

Text unit	Hash value
You can insert a 9 volt battery in the clock radio.	5
Should there be a power failure, the clock will continue to function.	11
You will not see this.	3
However, because the battery does not illuminate the clock display.	2
As soon as the power is switched on again, the display will indicate the correct time.	4

Fig. 1. Original Text of the Example

³ We currently use a local database generated from the 1911 edition of Roget’s Thesaurus [13], but we intend modify our system to use WordNet [10] instead.

In order to test the fingerprinting mechanism, we have defined two users with the following keys:

User	Key
user1	[2 – 6], [10 – 12], [4 – 8], [8 – 12]
user2	[0 – 2], [4 – 8], [10 – 12]

No seed is used in this example, which means that the fingerprints are embedded sequentially from the beginning of the text, this is necessary because the text is very short.

Text unit	Hash value
You can enter a 9 volt battery in the clock radio.	3
Should there be a power failure, the clock will continue to function.	11
You will not realize this.	5
However, because the battery does not light up the clock display.	9
As soon as the power is switched on again, the display will indicate the correct time.	4

Fig. 2. Example fingerprinted with key of user1

The text fingerprinted with the key of user1 is shown on Figure 2. The substitution of “enter” for “insert” in sentence 1 conveys the same meaning, but ‘the fingerprinted sentence sounds slightly artificial. Extending synonym substitution with semantic analysis capabilities, would allow the sentence to be fingerprinted as “A 9 volt battery may be inserted into the clock radio”, this phrase sounds more natural.

Text unit	Hash value
You may insert a 9 volt battery in the clock radio.	0
Should there be a power failure, the clock will carry on to function.	5
You will not witness this.	12
However, because the battery does not illuminate the clock display.	2
As soon as the power is switched on again, the display will indicate the correct time.	4

Fig. 3. Example fingerprinted with key of user2

The text fingerprinted with the key of user2 is shown on Figure 3. Again, the fingerprinted sentence conveys the same meaning as the original sentence. The replacement of “see” by “witness” in sentence 3, looks strange in a technical document. Again linguistic analysis and knowledge about the document domain might help improve the quality of the fingerprinting text.

The chosen text is difficult to fingerprint for the following reasons: the text unit is fairly short which gives fewer possibilities for synonym substitution (normal prose have longer sentences than the ones in our example), the nature of the document, a technical description, limits the natural vocabulary of the fingerprinted document and finally the subject of the text (a clock radio) was invented after the thesaurus that generated our database was published. We generally have better results with normal prose, but we expect similar results from larger technical documents, where the text unit may be increased to paragraph or sentence level.

4.2 Security Analysis

There are three principal kinds of attacks against the fingerprinting system: subtractive attack, distortive attack and additive attack [7]. In the following we present an informal analysis of these attacks.

Subtractive Attacks. A subtractive attack attempts to detect the location of the fingerprint and eliminate those sentences that carry the fingerprint. The attack is effective if the resulting document retains value to the attacker.

This attack depends on the user’s ability to identify the fingerprint within the document. With synonym substitution, a single user holding a single document cannot detect the fingerprint.

Two users with one document each can collude to detect those text units of the fingerprints where the intervals in their keys are different. Introducing redundancy in the keys would allow colluding users to be identified [6]. However, treason is normally best committed in secret which reduces the risk of collusion among users.

A single user, who receive multiple documents fingerprinted using the same key, will be able to identify text units with similar hash-values in all documents. Hash values are considered similar if their numerical distance is small compared to the overall width of the key. The identified text units form a superset of the fingerprinted text units, because two similar hash values may actually belong to neighbouring intervals, i.e., not be part of the key. The superset will converge towards the set of fingerprinted text units when more documents are considered. We therefore propose that new seeds are added to the key periodically. The length of this period depends, not only on the amount of documents delivered, but also on the length and the width of the key.

Distortive Attacks. A distortive attack attempts to modify the text in a way that distorts the fingerprint. The ability to detect the location of the fingerprint, reduce the amount of text that has to be modified, but overall degradation of the document is acceptable. The attack is effective if (part of) the fingerprint is modified and the document retains value to the attacker.

The ability to locate the fingerprint within the document can be limited by redundancy and periodically changing the seed in the user's key as described above. This means that a distortive attack has to rely on random transformations of the document in order to overwrite the fingerprint. In a document with 100 text units and a key length of 5, an attacker must perform 37 transformation in order to have more than 90% probability of overwriting one of the text units carrying the fingerprint. This shows that an attacker will have to modify a substantial part of the document in order to overwrite the fingerprint. Every transformation may have an insignificant impact on the overall message of the document, but the impact of the sum of these many transformations is likely to significantly distort the message of the document, i.e., the imperfection of textual transformations adds to the robustness of the synonym substitution. However, distortive attacks are viable if the length of the document is short compared to the length of the keys.

Additive Attacks. An additive attack inserts one or more extra fingerprints in the document, in order to mask the identity of the source of the leak.

Without knowledge of the keys, the attacker is unable to construct a valid fingerprint, he is therefore unable to launch an additive attack.

4.3 Limits of Synonym Substitution

It is inappropriate to apply semantic fingerprints to contracts, poems and other documents where the precise choice of words is decisive in determining the message. However, there exist many classes of documents, where the same message is conveyed by different choices of words, e.g., manuals (excluding technical terms), business letters, memos and informal instructions.

4.4 Information Hiding with Synonym Substitution

In this paper we focused on the use of synonym substitution to fingerprint documents in order to detect the source of leaked documents (traitor tracing). However, the mechanism presented here can also be used to hide information in a document.

A binary string can be embedded in a document in the following way. A key is chosen with the same length as the binary string. The first text unit selected by the key is transformed so that its hash value falls inside the key's interval if the first bit of the binary string is 1; otherwise the hash of the text unit must fall outside the interval. This procedure is repeated for all the bits in the binary string, so that the i 'th hash value falls inside the interval if the bit is 1 and outside the interval if the bit is 0.

5 Conclusions

In this paper we addressed the problems of fingerprinting documents encoded in modern logical markup languages, such as HTML and XML. These formats are quickly replacing traditional physical markup languages like Portable Document Format (PDF) and PostScript.

Traditional watermarking and fingerprinting techniques are based on imperceptible variations in the physical presentation of the document, which is no longer possible because logical markup leaves the rendering of the document to the user's software (browser). This means that new watermarking and fingerprinting techniques have to be developed.

We present synonym substitution, a technique for fingerprinting documents in logical markup based on semantics preserving transformations of the document's text. Synonym substitution is evaluated through a prototype developed for traitor tracing. We especially focus on traitor tracing, because this problem is simpler than the general fingerprinting problem. To the best of our knowledge, this work on synonym substitution is the worlds first presentation of a working semantic watermarking system.

Our experiments show that synonym substitution works, but that simple substitution requires intervention by the user who is fingerprinting the document. In order to progress towards a fully automatic system, we need to consider the context of the substitutions in the document. Techniques developed in the areas of natural language processing and machine translation may provide such context. Another area that has to be addressed is the identification of more safe transformations.

The ideas presented in this paper are part of an ongoing effort to develop a complete and automatic traitor tracing system based on semantic fingerprinting through synonym substitution.

The theory behind semantic watermarking combines theories from linguistics, electronic document processing and information hiding. Our experiments with synonym substitution provides some insight into this area, but much research remains to be done.

Acknowledgements. We would like to thank our colleague Carl Vogel for valuable input regarding the linguistic properties of synonym substitution. We would also like to thank Robert Kenny, David Whelan and Michelle Farrelly who contributed to various parts of the fingerprinting prototype. Finally, we would like to thank the anonymous reviewers for their comments, which helped improve the paper.

References

1. W. Bender, d. Gruhl, N. Morimoto, and A. Lu. Techniques for data hiding. *IBM Systems Journal*, 35(3&4), 1996.
2. J. T. Brassil, S. H. Low, N. F. Maxemchuk, and L. O'Gorman. Electronic marking and identification techniques to discourage document copying. In *Proceedings of Infocom*, pages 1278–1287, Toronto, Ontario, Canada, June 1994.
3. T. Bray, J. Paoli, C. M. Sperberg-McQueen, and E. Maler. *Extensible Markup Language (XML) 1.0 (Second Edition)*. 2000.
4. S. Brin, J. Davis, and H. Garcia-Molina. Copy detection mechanisms for digital documents. In *Proceedings of the ACM SIGMOD Annual Conference*, San Francisco, California, U.S.A., May 1995.
5. Andrei Z. Broder, Steven C. Glassman, Mark S. Manasse, and Geoffrey Zweig. Syntactic clustering of the web. SRC Technical Note 1997-015, DEC Systems Research Center (now COMPAQ), July 1997.

6. B. Chor, A. Fiat, and M. Naor. Tracing traitors. In *Advances in Cryptology—CRYPTO '94*, volume 839 of *Lecture Notes in Computer Science*, pages 257–270. Springer Verlag, 1994.
7. C. S. Collberg and C. Thomborson. Software watermarking: Models and dynamic embeddings. In *ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, San Antonio, Texas, U.S.A., January 1999.
8. J. H. Coombs, A. H. Renear, and S. J. DeRose. Markup systems and the future of scholarly text processing. *Communications of the ACM*, 30(11):933–947, November 1987.
9. S. J. DeRose, D. G. Durand, E. Mylonas, and A. H. Renear. What is text, really? *Journal of Computing in Higher Education*, 1(2):3–26, 1990.
10. C. Fellbaum, editor. *WordNet: An Electronic Lexical Database*. MIT Press, 1998.
11. Compris.com GmbH. Hide arbitrary data in texts through automatic text rephrasing with texthide! Available from: <http://www.TextHide.com/>.
12. Compris.com GmbH. Textsign – protect your texts with digital watermarks! Available from: <http://www.TextSign.com/>.
13. Project Gutenberg. *Roget's Thesaurus of English Words and Phrases*. The Crowell Company, 1911.
14. N. Heintze. Scalable document fingerprinting. In *Proceedings of the Second USENIX Workshop on Electronic Commerce*, Oakland, Ca, U.S.A., 1996.
15. S. H. Low and N. F. Maxemchuk. Performance comparison of two text marking and detection methods. *IEEE Journal on Selected Areas in Communications*, 16(4):561–572, May 1998.
16. S. H. Low and N. F. Maxemchuk. Capacity of text marking channel. *IEEE Signal Processing Letters*, December 2000. To appear.
17. S. H. Low, N. F. Maxemchuk, and A. Lapone. Document identification for copyright protection using centroid detection. *IEEE Transactions on Communications*, 46(3):372–383, March 1998.
18. N. F. Maxemchuk and S. H. Low. Marking text documents. In *Proceedings of International Conference on Image Processing*, Santa Barbara, California, U.S.A., 1997.
19. F. A. P. Petitcolas, R. J. Anderson, and M. G. Kuhn. Information hiding — a survey. *Proceedings of the IEEE, Special issue on protection of multimedia content*, 87(7):1062–1078, July 1999.
20. D. Raggett, A. Le Hors, and I. Jacobs. *HTML 4.01 Specification*. 1999.
21. V. Raskin, M. Atallah, C. McDonough, and S. Nirenburg. Natural language processing for information assurance and security an overview and implementation. In *Proceedings of the New Security Paradigms Workshop*, pages 51–65, Ballycotton, Ireland, September 2000.
22. N. Shivakumar and H. Garcia-Molina. Scam: A copy detection mechanism for digital documents. In *Proceedings of 2nd International Conference in Theory and Practice of Digital Libraries*, Austin, Texas, U.S.A., June 1995.
23. N. Shivakumar and H. Garcia-Molina. Building a scalable and accurate copy detection mechanism. In *Proceedings of 1st ACM Conference on Digital Libraries*, Bethesda, Maryland, U.S.A., March 1996.

SPEED Protocol: Smartcard-Based Payment with Encrypted Electronic Delivery^{*}

Antonio Ruiz¹, Gregorio Martínez¹, Oscar Cánovas², and Antonio F. Gómez¹

¹ Department of Information and Communications Engineering

² Department of Computer Engineering

University of Murcia, Spain

{arm, gregorio, skarmeta}@dif.um.es, ocanovas@dittec.um.es

Abstract. In these times of the dawning of e-commerce, many issues and barriers still remain to be solved before electronic transactions over the Web can be expected to be really successful. One important unresolved problem is the issue of having efficient and secure payment models based on e-purses and including electronic product delivery and price negotiation. In response to this need, the SPEED protocol specification has been proposed. This specification, which is described in this paper, provides a high level of security for all parties involved in e-commerce transactions over the Internet; at the same time, we have combined this aim with the use of highly-recognised standards and all the advantages of using e-purses implemented on multiapplication smart cards. Our work has also been tested in a real environment, providing us an interesting feedback based on technical and user-friendly matters.

1 Introduction

Nowadays, we are under a real revolution concerning the way people as consumers purchase the goods and services they need and desire. The web exceptional growth has created a new way for commercial transactions, called e-commerce, involving some powerful and interesting possibilities such as global market coverage and personalized and direct interaction with customers.

Although the needed technology is already in place, e-commerce has not really grown as expected. In fact, some people were predicting an incredible benefit for year 2000 but it was not so high. The main reason for this reluctant behaviour is that most customers still consider that on-line financial transactions over the Internet are not secure enough. Some other reasons are the lack of efficient distribution systems for delivering individual customer orders, and the existence of few national and no international laws concerning legal issues such as copyright, taxation, and on-line agreements and contracts.

These barriers have been observed by a number of companies and research institutions with interests in secure e-commerce markets. Their answer to this situation was to present a number of secure payment alternatives based on electronic

^{*} Partially supported by TEL-IFD97-1426 EU FEDER project (PISCIS)

money. Although some of these systems, such as PayWord [RS97], MicroMint [RS97], Millicent [G⁺95], NetBill [CTS95], etc. are secure and flexible enough, they have demonstrated to have many problems to reach real markets. The next list outlines some of the typical difficulties of most of the current proposals.

- The lack of pilot projects with real users
- Some proposals are not so light as intended
- There is no possibility for price negotiation (quite important in brokering markets)
- There is no perception of enough security by customers
- Electronic product delivery is not allowed
- There is no possibility of buying different products at the same time
- User mobility is not fully considered
- Some proposals are not based on recognised standards
- It is quite complicated to handle the change of electronic money
- The lack of proper arbitration (e.g., customer paid and did not receive the product)

Although some of these problems are already solved by some previous schemas, to provide a common answer to all them in the specific environment we are working, we have defined a new payment system called SPEED which is intended to provide a smart card based stored value payment schema to be used with electronic products delivery.

It is also indicating the scope of this protocol. In fact, e-purses schemas are normally used for frequent and/or fast payments with lower-medium value. So, we have defined and implemented an efficient and secure schema to buy some products such as music, newspapers, keys providing access to pay-per-view shows (like movies and sports) etc.

This basic definition is complemented with some other characteristics of high interest. We have designed this protocol with a negotiation phase (that it can be omitted if not needed) really important in brokering markets, encrypted delivery, and using recognised standards like ASN.1 [ITU95] and PKCS#7 [RSA93].

Furthermore, every player in our system (customer, vendor, and broker) has a private key, normally stored in his or her smart card (it is mandatory for customers and depends on efficiency issues for vendors and brokers), and a X.509 [HFS99] certificate issued by a supporting public key infrastructure. This security information is considered as the trust foundation of the whole payment system.

The reminder of this paper is organised as follows. Section 2 describes the type of smart cards and e-purses we are using in SPEED. Section 3 discusses the SPEED protocol, while Section 4 describes its messages and existing operation modes. Section 5 discusses main security issues concerning this protocol. Section 6 describes the pilot project where SPEED has been defined and its currently used, and outlines some performance analysis. Finally, some conclusions and future work are provided.

2 Smart Card with E-Purse as a Basic Component of SPEED

Memory card based e-purses today seem to be only suitable for closed systems where the e-purse provider is also the service provider or where there is a very low fraud incentive. The reason is that defrauding such systems is relatively easy. By interposing itself between an actual card and a point of payment, a small computer may record the secrets communicated during an initial transaction and can then, as often as required, be used to play the role of a card having the initial balance and to pay as the real user. This is the main reason why we decide to rely our system on smart cards with e-purses.

In our case, the e-purse is implemented as a multicurrency and multiapplication pre-paid IC (Integrated Circuit) card containing digital money which the cardholder can spend in return for goods and services (such as music, books, newspapers, and so on) from retailers. This money is decremented using some specific SAM (Secure Access Module) security modules, which are normally handled by banks. These modules have a daily record of every economic transaction performed on those electronic purses. Periodically, retailers are reimbursed with real money from banks, which already obtained the money in advance from cardholders at the time of storing the corresponding value in their cards (prepayment concept).

These electronic purses are based on the WG10 [\[CEN96,CEN95\]](#) (also called “Inter-sector Electronic Purse”) standard proposal defined by CEN (European Committee for Standardisation) which is made up of some of the most important European national standardisation bodies.

On the authentication process with the outside (i.e. the SAM modules) it is relevant to mention that our smart cards are able to run the 3DES symmetric cryptographic algorithm using prestored shared keys, as is described in the standard. A SAM security module uses the keys it shares with the card to authenticate messages during the decrement (i.e. purchase) or the increment transaction. This lets the SAM module convince the smart card that it is genuine. The card convinces the module by using the shared key to authenticate some information, like the amount to manage.

3 SPEED Overview

As commented before, the SPEED protocol is designed for purchasing and delivering electronically low and medium priced goods and services (MP3 songs, electronic documents, keys protecting video streams, etc.). The broker maintains accounts for vendors (and optionally for customers) and it hosts a set of SAM modules to perform decrement operations on the customer smart card e-purse. One SPEED transaction transfers electronic products from one vendor to one consumer, debiting the customer smart card e-purse (or account) and crediting the vendor account for the product price. SPEED is designed as a set of phases to support price negotiation, product delivery and payment.

There are two operation modes in the SPEED protocol: normal mode supports negotiation and it is designed with higher security capabilities (prevention of trivial denial of service attacks and full authentication of participating parties before the product delivery); the aggressive mode is composed by less messages than normal mode, and it is suitable for smaller products or scenarios with lower security requirements.

3.1 Players

The SPEED transaction model involves three main parties: the customer, the vendor, and the broker. Goods and services are delivered over the network, and SPEED links product delivery and payment into a single atomic transaction. The broker is not involved until the payment phase, when the customer submits a transaction request. Previously, the customer and the vendor can negotiate the product price in the negotiation phase.

Only registered users (customers and vendors) can interoperate using SPEED. Firstly, the customer and the vendor agree on the product to be purchased and its price. This can be accomplished after an optional offer and acceptance negotiation phase between the two parties. A customer receives the product he purchases if and only if he pays for the product. When the payment is performed, both customer and vendor obtain a proof of the transaction result (for example, for further complaints). The communication is protected from external entities using symmetric and, in some special cases, asymmetric cryptography.

SPEED assumes asymmetric trust relationships among the three participating entities. Brokers are assumed to be the most trustworthy, then vendors, and, finally, customers. Brokers play the role of serving as accounting intermediaries between customers and vendors. Vendors go into long-term relationships with brokers, in the same way as they would go into an agreement with a bank, credit card or company. Brokers tend to be large, well-known, and distinguished institutions or network service providers. The broker reputation is quite important for pulling toward customers; this reputation would be quickly loosed if customers or vendors have troubles with the broker. Vendor fraud consists of not providing correct products or descriptions. If this happens, customer will complain to the broker, and broker can drop vendors causing frequent complaints.

3.2 Registration

SPEED is designed considering that its participants are certified entities. It is assumed that both customers and vendors have a X.509v3 certificate issued by a trusted certification authority. The broker is also a certified authority, but it is not mandatory that the same certification authority certifies all the parties involved, although every certification authority must be considered trustworthy. Thus, a trusted CA must certify both customers and vendors before they initiate SPEED communications. As we will see below, real application environments

show that this is not a problem in order to use SPEED in electronic markets or similar scenarios.

Although the smart card e-purse is the basic payment model (and the preferred one by the authors of this paper) the use of bank accounts is also allowed in the SPEED protocol. In order to make use of this information in the payment process, customer should also provide this data in the registration process.

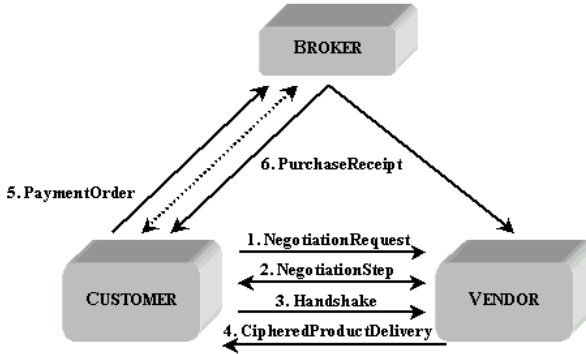


Fig. 1. SPEED messages in the normal operation mode

3.3 SPEED Purchase Overview

Figure 1 shows a global communication scheme conforming the product negotiation, delivery, and payment in the normal operation mode. We can see that messages 1, 2 and 3 constitute the product negotiation phase, and that they are exchanged between the customer and the vendor. Message 4 contains the product ciphred by a symmetric key generated by the vendor, which will be provided to the customer once the payment has been performed (messages 5 and 6). The broker and the customer will exchange additional messages if a smart card based payment has been selected as the payment mode (dotted line).

4 SPEED Protocol Specification

4.1 Notation

In this section we will describe the notation used to specify the sequence of messages conforming the SPEED protocol.

$[Data]$	It indicates that this piece of data is optional, and it could not be in the message.
$Hash(Data)$	A message digest of $Data$, obtained using a hash algorithm such as MD5 [Riv92].
$ Data _k$	$Data$, encrypted by a symmetric cipher using the key k .
$ Data ^k$	$Data$ is authenticated using a HMAC algorithm with a cryptographic key k . This represents a message composed by two elements: $Data$ and its cryptographic checksum.
$ Data _k^{k'}$	This is equivalent to $ Data ^k _k$
$\{Data\}_X$	$Data$, encrypted for X using public key cryptography (RSA). For computational efficiency, this is implemented using a digital envelop.
$\{Data\}_{X^{-1}}$	$Data$ is signed using the RSA private key of X .
$X \Rightarrow Y$	It indicates that X sends one message to Y .

In next sections we are going to describe all messages involved in the different phases. This explanation is divided in two blocks related to the normal and the aggressive operation modes.

4.2 Normal Mode

Generally, this operation mode is suitable in the next three scenarios:

- The customer does not know the product price because it is not fixed
- The customer wants to negotiate a lower price
- The product size, in bytes, is high (over 1 MB). As we will see in next sections, it could be very easy to perform a denial of service attack in the aggressive operation mode when the product size is high. For this reason, it is appropriate to use the normal mode if products are songs, videos, etc.

Negotiation Phase. This phase is composed of three messages: *NegotiationRequest*, *NegotiationStep* and *Handshake*. First two steps represent a request/response message pair where the customer requests or proposes a price, and the vendor replies to this price. Then, these steps may be repeated as needed until the customer and vendor agree on a price, or any of the two parties decide to break the negotiation. Finally, the *Handshake* message is sent by the customer to notify the vendor that the price is accepted.

The *NegotiationRequest* message is sent the first time a customer and vendor are negotiating a particular product. This message is digitally signed using the customer private key, and asymmetrically encrypted using the vendor public key.

$$1 \quad C \Rightarrow V \text{ NegotiationRequest } \{\{NID, SeqN, ProductID, [Price], \\ VendorID, EnKey, SignKey, Flag\}_{C^{-1}}\}_V$$

- *NID (Negotiation Identifier)*. It is a 4 bytes value generated by the customer using a pseudo-random number generator. The NID identifies the negotiation being performed between the customer and the vendor. Although this

identifier is not globally unique, it is used to distinguish between the different negotiations performed by the same vendor

- *SeqN* (*Sequence Number*). During the negotiation phase, it is possible to exchange several messages. Every message in this sequence must be unique in order to prevent reply attacks. For this reason, a *SeqN* field is present in the negotiation messages and each party must increment the *SeqN* value after receiving this type of messages
- *ProductID* (*Product Identifier*). It is a string composed by a product code and an application-specific description, which the vendor uses to specify the goods
- *Price*. It is the price that the customer may be willing to pay for the item. It is optional in this message because the customer may not know about the product price. Price is a string specifying a value and the used currency (coded using ISO 4217)
- *VendorID* (*Vendor Identifier*). This field is the digest of the vendor public key. It identifies the intended vendor of this negotiation request, and binds the negotiation request to a particular vendor. In this way, we try to avoid any possible customer impersonation from malicious vendors (we shall analyse this attack in section 5.4)
- *EnKey* (*Encryption Key*). *EnKey* is a symmetric key that will be used to provide confidentiality to the following messages exchanged between the customer and the vendor. It is a 128 bits long value generated by the customer. The default symmetric cipher to employ is IDEA [Lai92]
- *SignKey* (*Signing Key*). *SignKey* is a symmetric key used to provide integrity to the subsequent messages exchanged between the customer and the vendor. It is a 128 bits long value generated by the customer. The default cryptographic checksum function to employ is HMAC-MD5 [CG97]
- *Flag*. This field contains a boolean value indicating whether the proposed price is the last customer offer (i.e., the customer do not want to negotiate other price)

Some of these fields will appear later in this document, although we will explain them again only when they have a different meaning.

Both customers and vendors send *NegotiationStep* messages in order to negotiate the final price of the product. It is protected using symmetric cryptography and the keys previously exchanged (the message is authenticated using a HMAC algorithm and then ciphered using a symmetric cipher). In this message, each party makes a bid that could change in following messages (it depends on the negotiation strategy).

$$2 \quad V \Rightarrow C \text{ NegotiationStep } |NID, SeqN, Price, Flag|_{EnKey}^{SignKey}$$

- *Price* indicates the price proposed by the customer or vendor
- *Flag* codifies a boolean value that represents whether *Price* is the last offer

Both parties send the message in order to obtain the lowest (or the highest) product price. This exchange is repeated until one of the following conditions is reached:

1. The customer accepts the price proposed by the vendor in the last *NegotiationStep* message. In this case, the customer will send a *Handshake* message, as we will see below.
2. The customer or the vendor receive a *NegotiationStep* containing a last offer (flag is activated) that they do not agree with. In this case, the communication is closed.

Once the customer and the vendor have negotiated a price for the product, the *Handshake* message is sent to notify the vendor that the customer accepts his last offer. This message could have been sent after several negotiation steps (it indicates an active negotiation), or it could be the response to the first bid made by the vendor when the customer does not want to negotiate the price. The customer digitally signs the message (in this way the vendor cannot build a *Handshake* message using the shared secrets), and then this message is ciphered using *EnKey*.

$$3 \quad C \Rightarrow V \text{ Handshake } |\{NID, Price\}_{C^{-1}}|_{EnKey}$$

Product Delivery Phase. This phase is composed by one message named *CipheredProductDelivery*. When the vendor receives the *Handshake* message it proceeds to send the ciphered product. Then, the vendor constructs a message containing the product demanded by the customer, but ciphered with a pseudo-randomly generated symmetric key K . On the other hand, a bill is included in this message, which is digitally signed by the vendor, containing all the necessary information to uniquely identify this transaction. After that, this bill is ciphered using *EnKey*, and the message is sent to the customer.

$$4 \quad V \Rightarrow C \text{ CipheredProductDelivery } |Product|_K, |\{Bill\}_{V^{-1}}|_{EnKey}$$

- $|Product|_K$ is the ciphered product demanded by the customer. Once the payment is done, the symmetric key that protects this product will be sent to the customer in order to decipher the product.
- **Bill** = $\{AccountNumber, K\}_B, Hash(|Product|_K), Hash(ProductID), PaymentOrderID, Price, RecKey$
- *AccountNumber* is the vendor account number and K is the symmetric key ciphering the product. These fields are encrypted using the broker public key in order to maintain their confidentiality from the customer.
- $Hash(|Product|_K)$ is a message digest computed on the ciphered product, so the customer will detect any discrepancy before proceeding.
- $Hash(ProductID)$ is a message digest of the *ProductID* included in the first message. A hash (and not the complete identifier) is included in order to hide the product description from the broker. If the deciphered product did not match with the specified description, the customer would use this field to reclaim the correct product.
- *PaymentOrderID* is an identifier for the current transaction, and it must satisfy the condition of being unique in the whole system, considering all the transactions made by all the vendors. Two fields compose it: one is the message digest of the vendor public key, and the other is the NID value.

- *RecKey* (*Receipt Key*) is a symmetric key, 128 bits long, generated by the vendor. As we will see below, the broker will use this key to cipher the receipt.

Payment Phase. The number of messages composing this phase depends on the payment mode. There are two main messages, which are present in both operation modes. The *PaymentOrder* message is submitted by the customer to the broker, and marks the point of no return for the customer. Once the payment has been performed, using some of the available methods, the broker returns a receipt (the *PurchaseReceipt* message) including the symmetric key protecting the product, and some information that could be used for further complaints.

$$5 \quad C \Rightarrow B \text{ PaymentOrder } \{\{\{Bill\}_{V^{-1}}, \text{PaymentMode}\}_{C^{-1}}\}_B$$

- *Bill*. The customer includes in this message the bill previously received from the vendor in the *CipheredProductDelivery* message.
- *PaymentMode*. The structure of this field depends on the payment mode selected by the customer. We have the following options:
 - $\{AccountNumber\}$. If the customer selects the payment mode based on credit card number, this field represents the customer account number.
 - $\{BEncKey, BSignKey, S1\}$. As has been commented in previous sections, in order to perform a smart card based payment, it is needed to exchange a set of cryptographic messages between the card and a secure access module (hosted by the broker). Those two messages are part of the SPEED protocol, and therefore they must be protected like the other protocol messages. To do that, two symmetric keys are provided with the *PaymentOrder* message to authenticate (*BSignKey*) and to encrypt (*BEncKey*) the subsequent messages taking part of the decrement sequence. The *S1* field is necessary to authenticate the smart card and to do the payment.

When the smart card based payment mode is selected, two additional messages are needed in order to accomplish the payment. The content of these messages has been outlined in a previous section, and any further explanation will be omitted here.

$$\begin{aligned} a \quad & B \Rightarrow C \mid \text{PaymentOrderID}, S2 \mid_{BEncKey}^{BSignKey} \\ b \quad & C \Rightarrow B \mid \text{PaymentOrderID}, S3 \mid_{BEncKey}^{BSignKey} \end{aligned}$$

Once the broker decrements the quantity specified in the bill from the smart card or the account number, it proceeds to transmit to the customer and the vendor the receipt containing the product price, the encryption key and the *PaymentOrderID* concerning to this transaction. The broker digitally signs this message, and uses the *RecKey* contained in the bill to cipher it.

$$6 \quad B \Rightarrow C, V \text{ PurchaseReceipt } \{\{\text{PaymentOrderID}, \text{Price}, K\}_{B^{-1}}\}_{RecKey}$$

When the customer receives the message, he tries to decipher the product using the symmetric key *K*. If the key is wrong, the customer will use the receipt and the product description to make a complaint.

4.3 Aggressive Mode

In this operation mode, there is no negotiation phase since the customer accepts the price proposed by the vendor, and receives the bill and the ciphered product immediately. This protocol mode is suitable for scenarios where the negotiation has no sense (the price is fixed) or those situations where the number of transactions per second is the main goal. However, it is not suitable if the product size is large (over 1 MB) since it can be considered as a security flaw to perform denial of service attacks.

This mode reduces the number of messages (four) eliminating the *NegotiationStep* and *Handshake* messages, and modifying the *NegotiationRequest* and the *CipheredProductDelivery* messages. The rest of messages are the same that we have seen in the normal mode section.

Product Request Phase. The main differences between the *ProductRequest* message and the *NegotiationRequest* message (normal mode) are those fields related to the negotiation: the negotiation keys, the *flag* and the *SeqN* fields has been deleted since they are not necessities.

$$1 \quad C \Rightarrow V \text{ ProductRequest } \{\{NID, ProductID, Price, VendorID\}_{C^{-1}}\}_V$$

Product Delivery Phase. The main difference between the *CipheredProductDelivery* of the aggressive mode and the one of the normal mode is that the former bill is encrypted using the customer public key since there are not symmetric keys exchanged in previous phases. However, the message contents are the same.

$$2 \quad V \Rightarrow C \text{ CipheredProductDelivery } |Product|_K, \{\{Bill\}_{V^{-1}}\}_C$$

5 Security Analysis of the SPEED Protocol

This section provides a technical analysis of the cryptographic strength of the SPEED protocol, and it covers some other aspects related with vendor or customer frauds, complaints, and information visibility.

5.1 Assumptions about Cryptography

In general, our model assumes perfect cryptography. The following list explains what this assumption implies for all the cryptographic functions used in SPEED.

- **Opaque encryption.** Encryption is assumed to be opaque. If a message has the form $\{m\}_X$, only party X can learn m.
- **Unforgeable signatures.** Signatures are assumed to be unforgeable. Messages of the form $\{m\}_{X^{-1}}$ can only be generated by the party X. Anyone who has the verification key of X is able to verify that the message was indeed signed by X.

- **Collision-free hashes.** Hashes are assumed to be collision-free.
- **Trusted certificate authority.** There exists a trusted certificate authority (CA). All parties are assumed to have the verification key of the certification authority, and are able to verify messages signed by it.

5.2 General Objectives: Confidentiality and Authentication

The SPEED protocol encrypts all the information transmitted over the network. SPEED make use of both, asymmetric cryptography and symmetric encryption, in order to obtain a high performance in all transactions.

Symmetric encryption is used during the negotiation and the product delivery phases to protect the product and the bill, as well as in some other messages such as *PurchaseReceipt*. The rest of messages are encrypted using digital envelopes.

Short-term session keys are generated both for encryption and authentication, and they are transmitted using public key cryptography. 128 bits long keys and IDEA algorithm are used, which is currently considered strong enough to protect any private information. However, it is important that SPEED securely protect confidential data against even active attacks. Of course, underlying encryption algorithm should be secure against adaptive chosen plaintext / chosen ciphertext attacks, but this is not enough on its own. One important attack is cut-and-paste attack. SPEED uses the most comprehensive defense against this type of attacks, i.e., it uses strong authentication on all encrypted packets to prevent external modification of the ciphertext. Every message in the protocol is authenticated, and the way this is performed depends on the message, which can be authenticated by a digital signature or by a HMAC checksum.

5.3 Replay Attacks

The use of HMAC or digital signatures does not necessarily stop an adversary from replaying stale packets. There are some scenarios where this type of attack can obtain great benefits for an enemy:

- A malicious vendor could be interested in altering the contents of the negotiation messages exchanged between a particular customer and vendor. For example, the hostile vendor can capture the negotiation messages including the first price offered by the right vendor, and replay them later to the customer, preventing a possible handshake. This situation is solved inserting a negotiation message sequence number in every message of this phase. This number is incremented by each party before the message is transmitted. A negotiation message with a repeated sequence number is interpreted as an active attack, and it will cause the end of the communication.
- The customer receives more than once the same bill. With this replay attack, the vendor tries to swindle the costumer selling several times the same negotiated product. The customer can avoid this attack keeping track of every paid bill, which is identified by a unique *PaymentOrderID*. The same benefit could be obtained presenting previous bills to the broker in order to

force repeated payments. It is mandatory for broker to keep track of every processed transaction.

- A replay attack that could cause a denial of service is the repeated transmission of *Handshake* messages. If the vendor did not maintain a deliveries register, the attacker would force the reiterated transmission of large products, and therefore a performance loss of the attacked vendor. For this reason, vendors should keep track of every received *Handshake*.

5.4 Impersonation

Abadi and Needham postulated some basic engineering practices for cryptographic protocols in [AN96]. One of these principles is related to naming: “*if the identity of a principal is essential to the meaning of a message, it is prudent to mention the principals name explicitly in the message*”. Impersonation attack tries to convince some protocol party that the communication is being performed only between entities A and B, although there is a third party participating in that communication (impersonating A or B). We have included the *VendorID* field in the *NegotiationRequest* and *ProductRequest* messages in order to avoid the following situation (M is the malicious vendor):

- 1 $C \Rightarrow M$ *NegotiationRequest* $\{\{NID, SeqN, ProductID, Keys, Flag\}_{C^{-1}}\}_M$
- 2 $M \Rightarrow V$ *NegotiationRequest* $\{\{NID, SeqN, ProductID, Keys, Flag\}_{C^{-1}}\}_V$
- 3 $C \Leftrightarrow M \Leftrightarrow V$ *NegotiationStep* $|NID, SeqN, Price, Flag|_{EnKey}^{SignKey}$

In this scenario, the malicious vendor gains access to all the information exchanged during the negotiation phase (can learn the negotiation strategy of both parties, the product price, etc.), the customer ignores that he is not talking with the right vendor, and the honest vendor does not know that there is one man-in-the-middle. Using the *VendorID* field (an identifier of the intended vendor), any forwarded *NegotiationRequest* message can be interpreted as an attack.

5.5 Visibility

In a payment protocol, some of the exchanged data must be readable by only those parties needing this information to accomplish their tasks. For example, vendor account number should be protected from the customer, and vice versa, the product description should be hidden from the broker, etc. The SPEED protocol uses encryption and cryptographic checksums to protect this partially-confidential information from unintended readers. Vendor account number (contained in the bill) is encrypted using the broker public key; only a digest of the product description is inserted in the bill in order to hide it from the broker; and the customer never sends his account number (or some other information related with the smart card) to the vendor.

5.6 Product Delivery Attack

The information contained in the bill transmitted to the customer by the vendor can be different from the negotiated previously. Particularly, the vendor can be interested in changing the delivered product, or in modifying the price in order to get benefit from this change. The customer immediately detects any change in the product description (its hash) or in the product price, since the customer knows the product description (included in the first message) and its negotiated price (it is included in the *Handshake* message). Moreover, as we will see below, if the delivered product did not correspond with the provided description, the customer would possess all the information needed to demonstrate the fraud.

5.7 Customer Complaints

Despite the security mechanisms provided by the protocol, there are frauds that cannot be controlled as, for instance, the delivery of a product that does not match with its description. In this type of situations, the customer will make a complaint to obtain the required product.

- The delivered product does not match with its description. The customer does not receive the key protecting the product until the payment is done. Once the customer recovers this key, he deciphers the product and checks whether it matches with its description. If it does not match, he can present to the broker the product description exchanged in the first message (the broker has only the digest), and the deciphered product. In this way, the customer can demonstrate the vendor fraud.
- The customer does not receive the receipt containing the key. If some intruder had intercepted the *PurchaseReceipt* message, the customer would not be able to decipher the product. It is mandatory for the broker to maintain a register with the status of all the processed transactions in order to know whether it can retransmit the encryption key to the customer.

6 Using SPEED in a Real Environment: The PISCIS Project

The definition of projects and associated pilots on e-commerce marks a qualitative leap in the region or country where it is applied. In fact, we think that these projects are becoming more and more important to demonstrate final customers that related technology is mature enough.

In our case, the SPEED protocol has been designed and implemented as part of the PISCIS project. This project is intended to define an intelligent architecture to securely buy electronic products on the WEB. The communication base is a cable network provided by an important network service provider in our country.

To reach this objective, we have defined three main action lines as described now.

1. The design and implementation of a Java-based public key infrastructure, which provides all needed private keys and certificates to different players in our protocol: customers, vendors and broker (or brokers). The certification authority certificate, which it is distributed in off-line mode, is the trust foundation of the whole payment system
2. The design and implementation of a security module to access from final user applications to the information stored in smart cards. This module has been designed according to the Microsoft Windows security architecture [Cor01], developing a RSA Crypto Service Provider which implements the CryptoAPI 2.0
3. The design and implementation of the SPEED protocol, as described in this paper

The next subsection provides some more details on the implementation of the SPEED protocol in the environment defined by the PISCIS project.

6.1 SPEED Performance Analysis

We carried out the experiments using an Ethernet local area network. Computers running the customer software were Intel Pentium III 800 MHz processors with 128-MB main memory. Broker and server programs were run in Pentium II 450 MHz processors with 128-MB main memory. Windows 2000 professional edition was used as the operating system, and C++ language was used to develop the software. That software was compiled using the Visual C++ Studio 6.0 with the *maximize speed* optimization option activated.

SPEED was tested using product sizes of 16 bytes, 1MB, 3MB and 5MB. For each product size, we used up to five different computers simultaneously acting as customers. These customers buy products using the normal operation mode, and they accept the first price proposed by the vendor (only three messages constitute the negotiation phase). As Figure 2 shows, simultaneous customers are managed well. Purchase time is formed by two components: time concerning the product delivery, and time related to the rest of messages. The product delivery message originates the differences among those purchases of the same product size, but time used to exchange the rest of messages remains almost independent of the number of concurrent customers (as is shown by the results obtained from the purchase of the 16 bytes long product).

7 Conclusions

This paper has presented one smart-card based payment schema, called SPEED, for making small and medium price purchases over an open communication system. SPEED stands for Smartcard-based Payment with Encrypted Electronic Delivery. This name is providing us some of the main characteristics of this protocol: the use of a smart card electronic purse, the use of encryption as the main technique for protecting products, and the use of an electronic delivery method

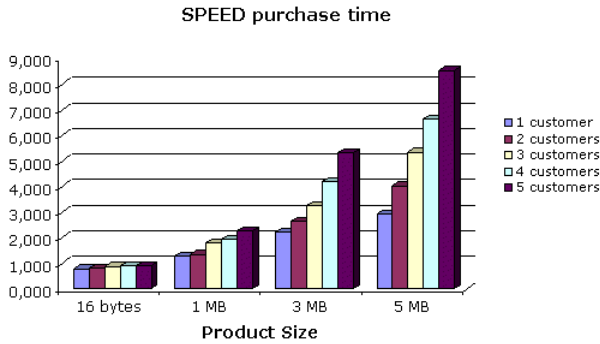


Fig. 2. SPEED purchase time

for sending products from merchants to customers. Some other important characteristics are also coming to our design, such as, security, non-repudiation, user-friendly, price negotiation of one product (or even several ones simultaneously) between the two speakers, web-based implementation, and the use of several standard formats (ASN.1, PKCS#7, X.509, etc).

Our work is really intended to fill the lack of proposals in the line of electronic payment methods based on smart card e-purses, defining a secure schema able to perform a high-speed price negotiation.

We have this payment system running from some time ago in an e-commerce pilot defined together with an important cable network company. As we have shown, results are really hopeful, getting the possibility of completing 4 or 5 transactions per second (with just one medium PC server) when the customer is buying a key to access to one real-time stream of information (music, movies, football matches, etc.)

8 Future Work

The focus of our future research activity is based on payments by mobile phone. In fact, we think that electronic commerce by mobile devices or m-commerce is predicted to be the next step in e-commerce, making use of mobile phones or PDAs with networking capabilities as the easiest, most flexible, and widespread smart card based operating platform. In this line, we are planning the adaptation of the SPEED protocol to m-commerce scenarios making use of some new research lines like SIM Application Toolkit with Java Cards [Sun00], WIN modules [WAP00], and so on.

We are also planning to make use of SPKI [EFL+99] credentials in our protocol to prove group membership in order to ask for special discounts to merchants.

Finally, we are also working in a new e-purse standard called CEPS [CEP99] (Common Electronic Purse Specifications), which it is going to provide interoperability between different standard proposals. It is based on a core set of common technologies and functions, allowing e-purse to operate in a homogeneous and global way.

References

- [AN96] M. Abadi and R. Needham. Prudent engineering practice for cryptographic protocols. *IEEE Transactions on Software Engineering*, 1(22):6–15, January 1996.
- [CEN95] CEN/TC224/WG10. *Inter-sector Electronic Purse, Part 3: Data Elements for Interchanges*, December 1995.
- [CEN96] CEN/TC224/WG10. *Inter-sector Electronic Purse, Part 2: Security Architecture*, January 1996.
- [CEP99] CEPSCO LLC. *Common Electronic Purse Specifications*, March 1999.
- [CG97] P. Cheng and R. Glenn. *Tests Cases for HMAC-MD5 and HMAC-SHA-1*, September 1997. Request For Comments (RFC) 2202.
- [Cor01] Microsoft Corporation. *CryptoAPI version 2.0*. World Wide Web, <http://msdn.microsoft.com/library/psdk/crypto>, 2001.
- [CTS95] B. Cox, J. D. Tygar, and M. Sirbu. Netbill security and transaction protocol. In *Proceedings of First USENIX Workshop on Electronic Commerce*, 1995.
- [EFL⁺99] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen. *SPKI certificate theory*, September 1999. Request For Comments (RFC) 2693.
- [G⁺95] S. Glassman et al. The Millicent protocol for inexpensive electronic commerce. *World Wide Web Journal, Fourth International World Wide Web Conference Proceedings*, pages 603–618, December 1995.
- [HFS99] R. Housley, W. Ford, and D. Solo. *Internet Public Key Infrastructure, Part I: X.509 Certificate and CRL Profile*, January 1999. Request for Comments (RFC) 2459.
- [ITU95] ITU-T. *ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)*, 1995. Recommendation X.690.
- [Lai92] X. Lai. *On the design and security of block ciphers*, volume 2. ETH Series in Information Processing, 1992.
- [Riv92] R. L. Rivest. *The MD5 Message-Digest Algorithm*, April 1992. Request For Comments (RFC) 1321.
- [RS97] R. L. Rivest and A. Shamir. Payword and MicroMint: two simple micropayment schemes. In Mark Lomas, editor, *Proceedings of 1996 International Workshop on Security Protocols*, number 1189 in Lecture Notes in Computer Science, pages 69–87. Springer, 1997.
- [RSA93] RSA Laboratories,. *PKCS#7: Cryptographic Message Syntax Standard*, November 1993.
- [Sun00] Sun Microsystems. *JavaCard 2.1.1 Specifications*, May 2000.
- [WAP00] WAP Forum. *Wireless Application Protocol Identity Module Specification*, February 2000.

Efficient Transferable Cash with Group Signatures

Ik Rae Jeong, Dong Hoon Lee, and Jong In Lim

Center for Information Security Technologies(CIST),
Korea University, Seoul, Korea

jir@cist.korea.ac.kr, {donghlee, jilim}@tiger.korea.ac.kr

Abstract. Various electronic cash systems have been developed in the literature to provide the clients' anonymity. But most of them do not provide transferability. Though a few cash systems have been proposed to provide transferability additionally, those systems are not efficient or lack of functionality. In this paper, using group signatures we suggest an efficient coin-based transferable off-line electronic cash system which provides anonymity control and unlinkability.

1 Introduction

The participants within a simple electronic cash system consist of banks, clients, and merchants. Clients and merchants have accounts at banks. Real money is transferred from the client's account to the merchant's account by using three cryptographic protocols: a withdrawal protocol with which the client withdraws coins against his account at the bank, a payment protocol with which the client pays coins to the merchant, and a deposit protocol with which the merchant deposits coins to the bank. The system is called off-line if payments can be performed without contacting the bank, or on-line otherwise. To simulate the functionalities of the real coin, two important requirements of e-cash systems are *anonymity* and *unlinkability*. Anonymity means that the relationship between the client and his purchases must be untraceable by anyone. Unlinkability means that given two cash anyone cannot decide whether or not these two cash were spent by the same client. A coin-based anonymous electronic cash system [4,10] protects the privacy of clients by the use of a blind signature scheme [9]. I.e., coins are blind signatures by which anonymity and unlinkability are achieved.

Another important requirement is *anonymity control*. In case of *multiple spending or deposit*, the bank is able to determine which of the two, a client and a merchant, was the guilty party and further identify the bad guy. This can be achieved by sealing the identity of the client in a coin in a way such that if the coin is spent twice by the same client, the information in both instances is used to reveal the identity.

Another important requirement of e-cash systems is *transferability*. Transferability means that the received cash can be transferred to other users without contacting the bank or an authority, i.e., in off-line. In transferable cash systems

the identities of honest users of the cash transferred multiple times must be kept anonymous.

Many electronic cash systems have been developed to provide the users' anonymity and anonymity control. But cash of those systems are not transferable to the third party. Though a few cash systems have been suggested to provide transferability, these system are not efficient or lack of functionality, i.e., unlinkability.

To provide transferability, two methods were suggested: using dummy blind signatures [14] and using anonymous public keys [20,21]. In [14] a user must use one dummy signature whenever cash is transferred to other user. Dummy signatures are issued using a blind signature scheme to provide anonymity. The user may get several dummy signatures at once to reduce the communication overhead, but this increases the storage overhead. In [20] the user gets the license anonymously in opening his account at the bank. In a transfer protocol the user uses the anonymous public key which was generated with the license, so the scheme does not provide unlinkability. The user in the scheme proposed in [21] registers at the anonymity server and gets a anonymous public key from the anonymity server. In a transfer protocol the user uses the anonymous public key, so this scheme also does not provide unlinkability. To avoid linkability, the authors suggested that the user gets and uses another anonymous public keys, but this increases the communication overhead with the anonymity server. In [21] recovering the identity of the user is possible only with the anonymity server when double spending is detected.

In this paper, we develop the coin-based transferable off-line cash systems using a group signature scheme. In the literatures, group signatures were applied to hide which bank issued a specific coin in multiple bank model [18,19]. The distinct feature of the proposed scheme is that group signatures are used to transfer the cash anonymously. To provide the anonymity revocation of double spenders, cash must contain the information of the spenders. To provide both the anonymity of honest spenders and unlinkability, this information also has to be anonymous and unlinkable. In our scheme the information of the spenders is contained in cash using a blind signature and group signatures. The blind signature is anonymous and unlinkable, and group signatures are also anonymous and unlinkable. So our scheme is anonymous and unlinkable. The scheme in [14] also provides both the anonymity of honest spenders and unlinkability using dummy signatures. But Since using dummy signatures needs the communication and storage overhead, our scheme is more efficient than the scheme in [14].

The outline of the paper is as follows. In Section 2, we describe the coin-based transferable off-line cash model and the components of the cash in the model. In Section 3, we define building blocks used in our scheme. In Section 4, We implement our coin-based transferable off-line cash system using group signatures. In Section 5 we mention how to extend our scheme to a fair cash system. Section 6 concludes the paper.

2 Model of Transferable Cash

Definition 1. The coin-based transferable off-line electronic cash system consists of the following procedures:

- **ECSet**: a key setup algorithm that outputs the secret key x_B and the public key y_B for the bank.
- **ECReg**: an interactive protocol (ECReg-B, ECReg-U) between the bank and a user. The user opens his account at the bank.
- **ECWith**: an interactive protocol (ECWith-B, ECWith-U) between the bank and a user. The user withdraws his money from his account at the bank to get the coin.
- **ECFTr**: an interactive protocol (ECFTr-S, ECFTr-R) between users. The user who withdraws his money from the bank transfers his coin to another user.
- **ECTr**: an interactive protocol (ECTr-S, ECTr-R) between users. The user with the transferred cash transfers it again to other user.
- **ECDep**: an interactive protocol (ECDep-U, ECTr-B) between a user and the bank. The user deposits his cash to the bank.

The coin-based transferable off-line electronic cash system must satisfy the unforgeability, the anonymity and unlinkability of honest users, the anonymity revocation of multiple transferers.

Definition 2. The coin-based transferable cash is the proof of transition of ownership of the coin. The transferred cash from the user u_i to the user u_{i+1} contains the following ingredients:

- *CCoin*: This proves that the cash containing this coin is valid.
- *CPf-Chain_i*: This proves that the ownership of the withdrawn coin is transferred to the user u_i .
- *CPf-L_i*: This proves that the ownership of the coin is transferred from the user u_i to the user u_{i+1} .
- *CCom_{i+1}*: This is the commitment of the identity of the user u_{i+1} . This is necessary to provide the revocation of the identities of the multiple transferers by the users or by the bank.
- *CPf-Com_{i+1}*: If *CCom_{i+1}* exists, this proves that *CCom_{i+1}* is the valid commitment of the user identity. If *CCom_{i+1}* does not exist, this is the certificate of the commitment of his identity. This is verified when the user u_{i+1} transfers this cash to the next user or deposits it to the bank.

Claim. In the coin-based off-line electronic cash system the cash must contain the above ingredients to provide unforgeability, anonymity and unlinkability of honest users, the anonymity revocation of multiple transferers, and transferability.

Sketch of proof. To provide unforgeability of the cash in the coin-based cash system, the cash must contain *CCoin* which is the coin which can be made only

by the bank. To prove that the coin was transferred to the previous owner, *CPf-Chain* is necessary. To prove that the cash was transferred to the current owner from the previous one, the proof *CPf-L* of the transition of the ownership is necessary. To prevent the others except the owner to transfer the cash, *CPf-L* can be made only by the owner of the cash. To revoke the anonymity of the current owner, when the current owner transfers this cash multiple times, the cash must keep the information about the owner, which is *CCom*. And to prove that *CCom* really contains the valid identification information of the owner, *CPf-Com* is also necessary. By *CCom* and *CPf-Com*, the anonymity of honest users and the anonymity revocation of multiple transferers are provided.

3 Primitives

Definition 3. A signature is a blind signature if the signer S issues a signature to a receiver R such a way that the view of the signer $View_S$ is independent of the message-signature pair $(m, BS_S(m))$. The blind signature system consists of the following procedures:

- **BSet**: a key setup algorithm that outputs a group B of order $|B|$, the secret key x_S and the public key y_S for the signer.
- **VBSign**: an interactive protocol (**BSign-S**, **BSign-R**) between the signer S and the receiver R . S issues a blind signature $BS_S(m)$ to R . **BSign-R** on input (m, y_S) blinds m with the blinding function f_b and sends $f_b(m)$ to **BSign-S**. **BSign-R** gets $BS_S(f_b(m))$ through the interaction with **BSign-S** on input (x_S, y_S) . **BSign-R** unblinds $BS_S(f_b(m))$ with the unblinding function f_u and outputs $BS_S((m)) = f_u(BS_S(f_b(m)))$.
- **BVer**: a deterministic algorithm that on input S 's public key y_S and $(m, BS_S(m))$ outputs 1 if and only if $(m, BS_S(m))$ is a valid message-signature pair.

The blind signature schemes such as that in [4] can be used in our scheme, in which anonymity revocation of the double spender is possible.

Definition 4. A signature based on a proof of knowledge of representations of y_1, \dots, y_w with respect to the bases g_1, \dots, g_v on the message m is denoted as follows :

$$SPK[(\alpha_1, \dots, \alpha_u) : (y_1 = \prod_{j=1}^{l_1} g_{b_{1j}}^{\alpha_{e_{1j}}} \wedge \dots \wedge (y_w = \prod_{j=1}^{l_w} g_{b_{wj}}^{\alpha_{e_{wj}}})](m),$$

where the indices $e_{ij} \in \{1, \dots, u\}$ refer to the elements $\alpha_1, \dots, \alpha_u$, the indices $b_{ij} \in \{1, \dots, v\}$ refer to the base elements g_1, \dots, g_v and $l_i \in \{1, \dots, v\}$ is the number of the base elements of y_i . The signature consists of an $(u + 1)$ tuple $(c, s_1, \dots, s_u) \in \{0, 1\}^k \times Z_n^u$ satisfying the equation

$$c = H(m || y_1 || \dots || y_w || g_1 || \dots || g_v || y_1^c \prod_{j=1}^{l_1} g_{b_{1j}}^{s_{e_{1j}}} || \dots || y_w^c \prod_{j=1}^{l_w} g_{b_{wj}}^{s_{e_{wj}}}).$$

For example, the discrete logarithm of y_1 to the base g_2 is α_1 satisfying $y_1 = g_2^{\alpha_1}$ and the representation of y_2 to the bases (g_1, g_2) is (α_1, α_2) satisfying $y_2 = g_1^{\alpha_1} g_2^{\alpha_2}$. Then $SPK[(\alpha_1, \alpha_2) : y_1 = g_2^{\alpha_1} \wedge y_2 = g_1^{\alpha_1} g_2^{\alpha_2}](m)$ is used for proving the knowledge of the discrete logarithm of y_1 and the representation of y_2 and that the g_1 part of this representation is equal to the discrete logarithm of y_1 without revealing any information about (α_1, α_2) . This SPK can be computed only if the secrets (α_1, α_2) are known.

Definition 5. A signature based on a proof of knowledge of representations of y_1, \dots, y_w with respect to the bases g_1, \dots, g_v on the message m with random numbers $\gamma_1, \dots, \gamma_u$ which are committed to $(\prod_{j=1}^{l_1} g_{b_{1j}}^{\gamma_{e_{1j}}}) \wedge \dots \wedge (\prod_{j=1}^{l_w} g_{b_{wj}}^{\gamma_{e_{wj}}})$ is denoted as follows :

$$CR-SPK < (\gamma_1, \dots, \gamma_u) : (\prod_{j=1}^{l_1} g_{b_{1j}}^{\gamma_{e_{1j}}}) \wedge \dots \wedge (\prod_{j=1}^{l_w} g_{b_{wj}}^{\gamma_{e_{wj}}}) > [(\alpha_1, \dots, \alpha_u) : (y_1 = \prod_{j=1}^{l_1} g_{b_{1j}}^{\alpha_{e_{1j}}}) \wedge \dots \wedge (y_w = \prod_{j=1}^{l_w} g_{b_{wj}}^{\alpha_{e_{wj}}})](m),$$

where the indices $e_{ij} \in \{1, \dots, u\}$ refer to the elements $\alpha_1, \dots, \alpha_u$, the indices $b_{ij} \in \{1, \dots, v\}$ refer to the base elements g_1, \dots, g_v and $l_i \in \{1, \dots, v\}$ is the number of the base elements of y_i . The signature consists of an $(u + 1)$ tuple $(c, s_1, \dots, s_u) \in \{0, 1\}^k \times Z_n^u$ satisfying the equation

$$\begin{aligned} c &= H(m || y_1 || \dots || y_w || g_1 || \dots || g_v || \prod_{j=1}^{l_1} g_{b_{1j}}^{\gamma_{e_{1j}}} || \dots || \prod_{j=1}^{l_w} g_{b_{wj}}^{\gamma_{e_{wj}}}) \\ &= H(m || y_1 || \dots || y_w || g_1 || \dots || g_v || y_1^c \prod_{j=1}^{l_1} g_{b_{1j}}^{s_{e_{1j}}} || \dots || y_w^c \prod_{j=1}^{l_w} g_{b_{wj}}^{s_{e_{wj}}}). \end{aligned}$$

For example, the discrete logarithm of y_1 to the base g_2 is α_1 satisfying $y_1 = g_2^{\alpha_1}$ and the representation of y_2 to the bases (g_1, g_2) is (α_1, α_2) satisfying $y_2 = g_1^{\alpha_1} g_2^{\alpha_2}$. Then $CR-SPK < (\gamma_1, \gamma_2) : g_2^{\gamma_1}, g_1^{\gamma_1} g_2^{\gamma_2} > [(\alpha_1, \alpha_2) : y_1 = g_2^{\alpha_1} \wedge y_2 = g_1^{\alpha_1} g_2^{\alpha_2}](m)$ is used for proving the knowledge of the discrete logarithms (α_1, α_2) with committed random numbers γ_1, γ_2 .

Definition 6. Let g be a generator of the group and y be the public key of the receiver R . An ElGamal encryption of m with the receiver's public key $y = g_x$ is $ElG_R(m) = (D_1, D_2)$ where

$$D_1 = g^k, D_2 = my^k.$$

The receiver can recover the message m from $ElG_R(m) = (D_1, D_2)$ with his secret key x as follows :

$$D_1^{-x} D_2 = g^{-kx} m g^{kx} = m.$$

In group signature schemes, each and only group member can sign the message with respect to the group public key, thus the signature is verifiable with the group public key (*unforgeability*). Even though the group manager and group

members collude, they can not generate a group signature which will be linked with a group member not in the coalition (*no framing*). Given a signature anyone except the group manager cannot identify the member who generated the signature (*anonymity*). Only the group manager can revoke the anonymity of the signer of the given signature, if needed (*revocability of anonymity*). Furthermore given two group signatures anyone except the group manager cannot tell whether these signatures were made by the same group member (*unlinkability*).

In our cash scheme we use the group signature scheme in which the group manager GM recovers the group member u 's identity by decrypting an ElGamal encryption. This is accomplished by a group signature $ElG-GS_u(m)$ generated by u .

Definition 7. The group signature scheme in our model consists of the following procedures:

- **GSet**: a key setup algorithm that outputs a group G of order $|G|$, a base g such that computing discrete logarithm is infeasible, and the group manager's secret key x_{GM} and public key y_{GM} .
- **GReg**: an interactive protocol (**GReg-GM**, **GReg-U**) between the group manager and the user. The user u gets the member's certificate from GM . **GReg-U** on input y_{GM} outputs his secret key x_u and the certificate v_u through the interaction with **GReg-GM** which has x_{GM} as input. Both **GReg-GM** and **GReg-U** output the certificate v_u and $GCom(x_u)$ which is the commitment of the member's secret key x_u . The commitment $GCom(x_u)$ and the certificate v_u are linked with the group member's identity u .
- **GSig**: a probabilistic algorithm that on input (x_u, v_u, y_{GM}) outputs the group signature

$$ElG-GS_u(m) = Proof[(x_u, v_u) : (D_1, D_2) = ElG_{GM}(ID_u) \wedge v_u = S_{GM}(GCom(x_u))](m),$$

where ID_u is used for identifying the group member and can be $GCom(x_u)$ or v_u . $ElG-GS_u(m)$ proves that the group member knows (x', v') which satisfies the predicate $(D_1, D_2) = ElG_{GM}(ID') \wedge v' = S_{GM}(GCom(x'))$ without revealing (x', v') .

- **GVer**: a deterministic algorithm that on input GM 's public key y_{GM} and $(m, ElG_{GM}(ID_u), ElG-GS_u(m))$ outputs 1 if and only if $(m, ElG_{GM}(ID_u), ElG-GS_u(m))$ is a valid message-signature pair.
- **GRev**: a deterministic algorithm that on input GM 's secret key x_{GM} and $(m, ElG_{GM}(ID_u), ElG-GS_u(m))$ outputs ID_u by decrypting $ElG_{GM}(ID_u)$, thus recover the identity of the member u for the given group signature.

The group signature scheme in [6] has the above signature structure. The group signature schemes in [18] with the similar signature structure can be used in our model.

4 Transferable Cash Using Group Signatures

In our scheme the ingredients of the i -times transferred cash is constructed as follows:

- $CCoin = \{m, BS_B(m)\}$.
- $CPf-Chain_1 = \{\}$;
 $CPf-Chain_i = \{CPf-L_1, CCom_2, CPf-Com_2, \dots, CPf-L_{i-1}, CCom_i, CPf-Com_i\}$, for $i > 1$.
- $CPf-L_1 = Proof[(R_{ID}) : m = BCom(ID_{u_1}, R_{ID})](ElG_{GM}(ID_{u_2}))$,
 where $BCom(ID_{u_1}, R_{ID})$ is the commitment of the user u_1 's identity with a random R_{ID} ;
- $CPf-L_i = CR-SPK < (b_i) : g^{b_i} > [(k_i) : D_1^i = g^{k_i}](ElG_{GM}(ID_{u_{i+1}}))$, for $i > 1$.
- $CCom_{i+1} = ElG_{GM}(ID_{u_{i+1}}) = \{D_1^{i+1} = g^{k_{i+1}}, D_2^{i+1} = ID_{u_{i+1}} y_{GM}^{k_{i+1}}\}$.
- $CPf-Com_{i+1} = (g^{b_{i+1}}, ElG-GS_{u_{i+1}}(g^{b_{i+1}}))$.

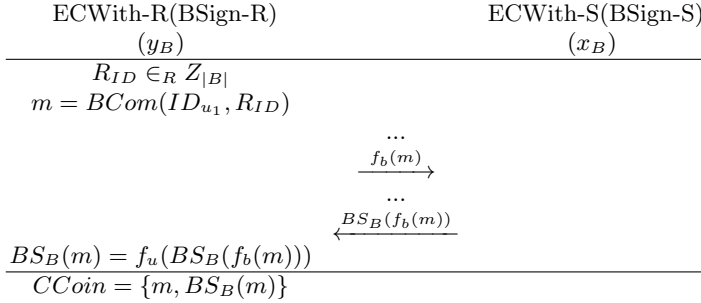


Fig. 1. Withdrawal Protocol(ECWith)

Our coin-based transferable off-line cash system is constructed with the following procedures:

- **Setup Stage:** the bank runs BSet and the group manager runs GSet to get their secret-public key pairs respectively.
- **Registration Stage:** the user u_i runs EReg-U to open his account. And he also runs GReg-U to get his secret key $x_{u_{i+1}}$, the commitment $GCom(x_{u_{i+1}})$ and the certificate v_{u_i} .
- **ECWith:** ECWith is BSign in our scheme. Thus the user runs BSign-R to get a coin $CCoin = \{m, BS_B(m)\}$ from the bank which runs BSign-S. The user makes the commitment m of his identity with a random R_{ID} , blinds the commitment using the blinding function f_b , and sends it to the bank. The bank signs the blinded commitment, and sends it back to the user. The user extracts the blind signature $BS_B(m)$ using the unblinding function f_u . ECWith is shown in Fig 1.

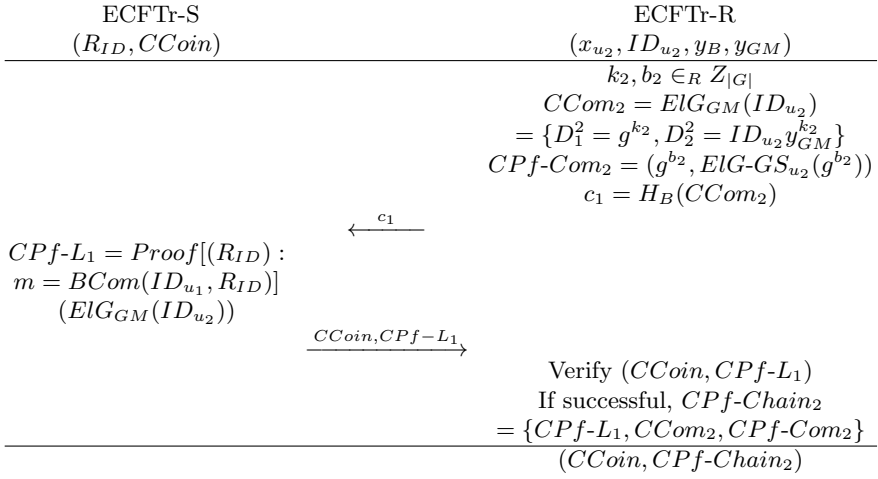
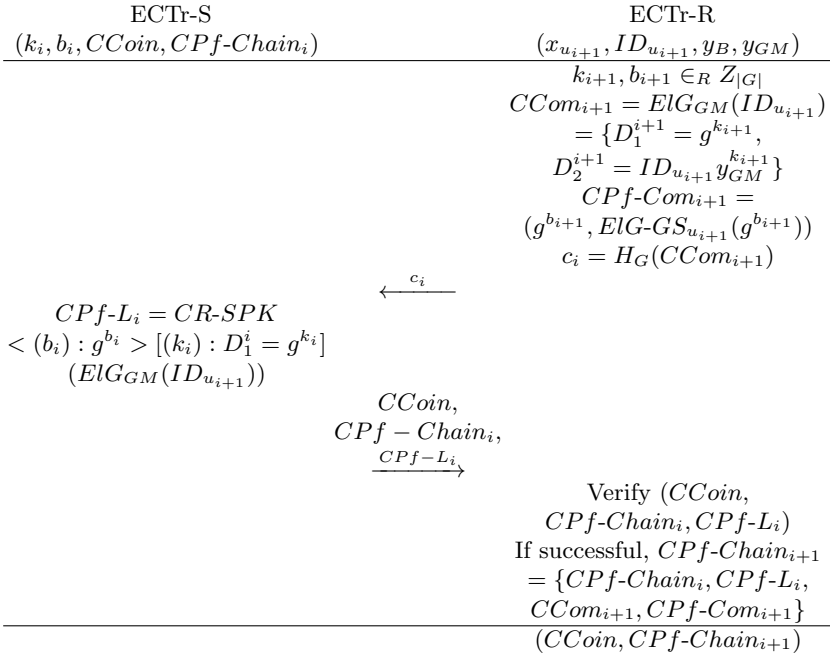


Fig. 2. First Transfer Protocol(ECFTr)

- ECFTr: The sender runs ECFTr-S on input $CCoin$ and the receiver runs ECFTr-R. The receiver makes an encryption of his identity $CCom_2$ and a proof of correctness for that encryption $CPf-Com_2$, and sends to the receiver the challenge c_1 using H_B which is the collision-resistant hash function used in the proof system $CPf-L_1$. The sender sends to the receiver the coin $CCoin$ and a proof $CPf-L_1$ using the challenge value, which proves that he is the legitimate owner of the coin, and by which the sender transfers the ownership of cash to the receiver. The receiver verifies the coin and the proof, and accepts if they are valid. ECFTr is shown in Fig 2.
- ECTr: The sender runs ECTr-S on input $(CCoin, CPf-Chain_i)$ and the receiver runs ECTr-R. The receiver makes an encryption of his identity $CCom_{i+1}$, a proof of correctness for that encryption $CPf-Com_{i+1}$, and sends to the receiver the challenge c_i using H_G which is the collision-resistant hash function used in the proof system $CPf-L_i$. The sender sends to the receiver the coin $CCoin$, the chain of proofs $CPf-Chain_i$ which proves that the coin was transferred to the user whose identity is encrypted into $CCom_i$, and a proof $CPf-L_i$ using the challenge value, which proves that he is the legitimate owner of cash, and by which the sender transfers the ownership of cash to the receiver. The receiver verifies the coin, the chain of proofs and the proof, and accepts if they are valid. ECTr is shown in Fig 3.
- ECDep: The user runs ECDep-U on input $(CCoin, CPf-Chain_i)$ and the bank runs ECDep-B. The user makes the challenge c_i with his identification information and the date by himself, and a proof $CPf-L_i$ using this challenge value which proves that he is the legitimate owner of cash. And then he sends the coin $CCoin$, the chain of proofs $CPf-Chain_i$ which proves that the coin was transferred to the user whose identity is encrypted into $CCom_i$, the

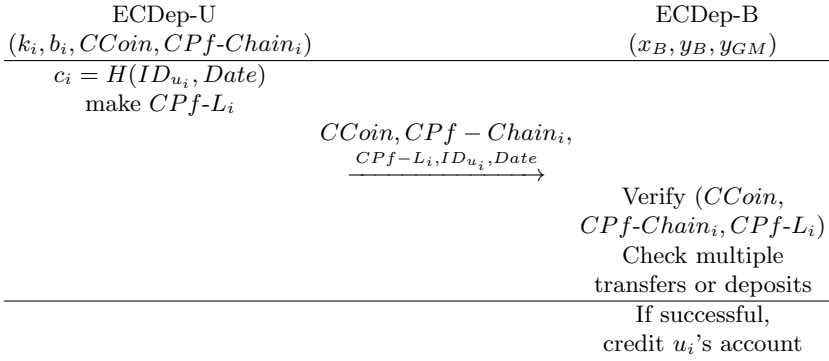
**Fig. 3.** Transfer Protocol(ETr)

proof $CPf-L_i$, his identification information ID_{u_i} , and the date $Date$ to the bank. The bank verifies the deposited cash and checks whether or not it is transferred or deposited multiple times. If it is valid and not used multiple times, the bank credits the user's account. ECDep is shown in Fig 4.

Theorem 1. *Our coin-based transferable off-line electronic cash system provides unforgeability, anonymity and unlinkability of honest users, and the anonymity revocation of multiple transferers.*

proof.

- *unforgeability.* On the assumption that BS_B , $CPf-L_1$, $CR-SPK$ and $ElG-GS_u$ are unforgeable and the hash functions H_B and H_G are collision-resistant, our scheme provides unforgeability. The i -times transferred cash is as follows: $Cash = \{CPf-L_1, CCom_2, CPf-Com_2, \dots, CPf-L_i, CCom_{i+1}, CPf-Com_{i+1}\}$. To forge the cash, the forger must forge at least one of BS_B , $CPf-L_1$, $CR-SPK$ and $ElG-GS_u$ or find the collision of the hash functions. By our assumption, it is impossible.
- *anonymity and unlinkability of honest users.* The cash is constructed using BS_B , $CPf-L_1$, $CR-SPK$, $ElG-GS_u$, $ElG_{GM}(ID_u)$. BS_B , $ElG-GS_u$ and $ElG_{GM}(ID_u)$ are anonymous and unlinkable by the assumptions. $CPf-L_1$

**Fig. 4.** Deposit Protocol(ECDep)

and $CR-SPK$ are also anonymous and unlinkable if the cash is not transferred multiple times. Thus our cash is anonymous and unlinkable if the cash is not transferred multiple times.

- *anonymity revocation of multiple transferers*: If the first user transfers the same cash multiple times, he has to make $CPf-L_1$ multiple times with different challenges. Then anyone can recover the identity of multiple transferer with BS_B and the instances of $CPf-L_1$ using the anonymity revocation mechanism of multiple transferers of the underlying blind signature. If the user u_i except the first user transfers the same cash multiple times, he has to make $CR-SPK$ multiple times with different challenges. Then anyone can recover the identity of u_i with $ElG_{GM}(ID_{u_i}), g^{b_i}, ElG-GS_{u_i}(g^{b_i})$ and the instances of $CR-SPK$ as follows:

$$\{ElG_{GM}(ID_{u_i}) = (D_1^i = g^{k_i}, D_2^i = ID_{u_i} y_{GM}^{k_i}), g^{b_i}, ElG-GS_{u_i}(g^{b_i})\}$$

$$CR-SPK < (b_i) : g^{b_i} > [(k_i) : D_1^i = g^{k_i}](ElG_{GM}(ID_{u_{i+1}})) = (c, s)$$

$$CR-SPK < (b_i) : g^{b_i} > [(k_i) : D_1^i = g^{k_i}](ElG_{GM}(ID_{u'_{i+1}})) = (c', s')$$

$$\frac{s-s'}{c'-c} = k_i$$

$$\frac{D_2^i}{y_{GM}^{k_i}} = ID_{u_i}.$$

5 Extension

A *fair* electronic cash system has one more participant, a *trustee(s)*, who could recover the identities of users only if certain conditions are met. Two revocability features are *cash tracing* and *users tracing*. In cash tracing, given the information of the withdrawal protocol provided by the bank, the trustee returns the information that can be used to identify the cash. Thus this feature defeats the blackmailing in the case a legitimate user is blackmailed. In users

tracing, the bank gives to the trustee the information of the cash. Then the trustee gives back to the bank the account IDs or the information of the withdrawal protocol instance that can be used to identify the users. Users tracing allows to prevent money laundering by tracing the users of suspicious cash. We develop the revocation mechanism for the first user and the others separately. To extend our scheme to the fair cash system, we must adopt ECWith and ECFTr from a non-transferable cash system which provides *coin tracing* and *user tracing*. Then we can provide cash tracing and users tracing as follows:

Cash Tracing. The trustee can trace the cash from the record of the withdrawal protocol which is kept in the bank using coin tracing mechanism of the underlying non-transferable cash system.

Users Tracing. If needed, the trustee and the group manager revoke the anonymity of the users. Given a *Cash*, the trustee recovers the first user's identity using user tracing mechanism of the underlying non-transferable cash system. The group manager can recover the users except the first user by decrypting the ElGamal encryptions in a *Cash*. Note that the trustee cannot recover the other users' identities except the first user while the group manager can know the identities of the users except the first user. In our scheme the group manager also plays a role of the trustee.

This model is also easily extendable to multiple bank model, where the anonymity control of banks also has to be considered. If banks form a group and issue cash by using a group signature scheme, the anonymity control of banks can be achieved as proposed in [18].

6 Conclusion

We suggested the coin-based transferable off-line electronic cash system using group signatures. Our scheme does not use dummy signatures, thus more efficient than those using dummy signatures [14]. Our model also provides unlinkability, thus the anonymities of the users are more assured than those in [20,21]. In the model the size of the coin increases when it is transferred. This is indispensable in coin-based transferable cash systems that provides anonymity control and proved in [14]. Our scheme is forward traceable, i.e., if a user sees the cash later in the chain of transferring, he can recognize it.

References

- [1] Giuseppe Ateniese, Jan Camenisch, Marc Joye, and Gene Tsudik. A Practical and Provably Secure Coalition-Resistant Group Signature Scheme. *Advances in Cryptology-CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 255-270, Springer Verlag, 2000.
- [2] G. Ateniese, M. Joye, and G. Tsudik. On the Difficulty of Coalition-Resistance in Group Signature Schemes. *Security in Computer Networks 1999*.

- [3] G. Ateniese and G. Tsudik. Some open issues and new directions in group signature. *Proceedings of the International Conference on Financial Cryptography*, volume 1648 of *Lecture Notes in Computer Science*, pages 196-211. Springer-Verlag, 1999.
- [4] S. Brands. Untraceable off-line cash in wallets with observers. *Advances in Cryptology - CRYPTO '93*, volume 773 of *Lecture Notes in Computer Science*, pages 302-318. Springer-Verlag, 1993.
- [5] E. Brickell, P. Gemmel, and D. Kravitz. Trustee-based tracing extensions to anonymous cash and the making of anonymous change. *Proceedings of the Sixth Annual Symposium on Discrete Algorithm*, pages 457-466. Association for Computing Machinery, Jan. 1995.
- [6] J. Camenish and M. Michels. A group signature scheme with improved efficiency. *Advances in Cryptology - ASIACRYPT '98*, volume 1514 of *Lecture Notes in Computer Science*, pages 160-174. Springer-Verlag, 1998.
- [7] J. Camenisch, U. Maurer, and M. Stadler. Digital payment systems with passive anonymity-revoking trustees. *Computer Security - ESORICS '96*, volume 1146 of *Lecture Notes in Computer Science*, pages 33-43. Springer-Verlag, 1996.
- [8] J. Camenisch and M. Stadler. Efficient group signature scheme for large groups. *Advances in Cryptology - CRYPTO '97*, volume 1296 of *Lecture Notes in Computer Science*, pages 410-424. Springer-Verlag, 1997.
- [9] D. Chaum. Blind signature systems. In D. Chaum, editor, *Advances in Cryptology-CRYPTO '83*, page 153. Plenum, 1983.
- [10] D. Chaum, A. Fiat, and M. Naor. Untraceable electronic cash. *Advances in Cryptology-CRYPTO '88*, volume 403 of *Lecture Notes in Computer Science*, pages 319-327. Springer-Verlag, 1990.
- [11] D. Chaum and T. Pedersen. Wallet databases with observers. *Advances in Cryptology-CRYPTO '92*, volume 740 of *Lecture Notes in Computer Science*, pages 89-105. Springer-Verlag, 1992.
- [12] David Chaum and Torben Pryds Pedersen. Transferred Cash Grows in Size. *Advances in Cryptology-EUROCRYPT '92*, volume 658 of *Lecture Notes in Computer Science*, pages 390-407. Springer-Verlag, 1992.
- [13] D. Chaum and E. van Heyst. Group signatures. *Advances in Cryptology-EUROCRYPT '91*, volume 547 of *Lecture Notes in Computer Science*, pages 257-265. Springer-Verlag, 1991.
- [14] L. Chen and T. P. Pedersen. New group signature schemes. *Advances in Cryptology-EUROCRYPT '94*, volume 950 of *Lecture Notes in Computer Science*, pages 171-181. Springer-Verlag, 1995.
- [15] A. de Solages and J. Traore. An efficient fair off-line electronic cash system with extensions to checks and wallets with observers. *Proceedings of the Second International Conference on Financial Cryptography*, volume 1465 of *Lecture Notes in Computer Science*, pages 275-297. Springer-Verlag, 1998.
- [16] Y. Frankel, Y. Tsiounis, and M. Yung. "Indirect discourse proofs": Achieving fair off-line e-cash. *Advances in Cryptology-ASIACRYPT '96*, volume 1163 of *Lecture Notes in Computer Science*, pages 286-300. Springer-Verlag, 1996.
- [17] Y. Frankel, Y. Tsiounis, and M. Yung. Fair off-line e-cash made easy. *Advances in Cryptology-ASIACRYPT '98*, volume 1514 of *Lecture Notes in Computer Science*, pages 257-270. Springer-Verlag, 1998.
- [18] Ik Rae Jeong and Dong Hoon Lee. Anonymity Control in Multi-bank e-Cash System. *Progress in Cryptology - INDOCRYPT 2000*, volume 1977 of *Lecture Notes in Computer Science*, pages 104-116. Springer-Verlag, 2000.

- [19] A. Lysyanskaya and Z. Ramzan. Group Blind Digital Signatures: A Scalable Solution to Electronic Cash, *Proceedings of the Second International Conference on Financial Cryptography*, volume 1465 of *Lecture Notes in Computer Science*, pages 184-197, Springer-Verlag, 1998.
- [20] T. Okamoto and K. Ohta. Universal electronic cash. *Advances in Cryptology-EUROCRYPT '91*, volume 547 of *Lecture Notes in Computer Science*, pages 324-337. Springer-Verlag, 1991.
- [21] H. Pagnia and R. Jansen. Towards Multiple-Payment Schemes for Digital Money. *FC '97 Financial Cryptography*, volume 1318 of *Lecture Notes in Computer Science*, pages 203-216. Springer-Verlag, 1997.
- [22] M. Stadler, J. M. Piveteau, and J. Camenisch. Fair blind signatures. *Advances in Cryptology-EUROCRYPT '95*, volume 921 of *Lecture Notes in Computer Science*, pages 209-219. Springer-Verlag, 1995.
- [23] S. von Solms and D. Naccache. On blind signatures and perfect crimes. *Computer & Security*, volume 11, pages 581-583, 1992.

An Auditable Metering Scheme for Web Advertisement Applications

Liqun Chen and Wenbo Mao

Hewlett-Packard Laboratories, Bristol,
Filton Road, Stoke Gifford, Bristol BS34 8QZ, UK
{Liqun.Chen, Wenbo.Mao}@hp.com

Abstract. This paper proposes a cryptographic scheme for metering the duration and/or the number of instances of running a process. This scheme has the following property: knowing a secret, one can validate a piece of metering evidence in constant units of time while without the secret the job of generating a valid piece of evidence requires time indicated by the evidence. Because the scheme utilises a well-known computational complexity problem, the meter based on it can be implemented in software yet is tamper-resistant.

We will address the use of this scheme in building an auditable metering scheme for finding the popularity of web sites. The scheme is suitable for rapidly increasing web advertisement applications. We will also discuss the related security issues.

Keywords. Auditable metering, Web advertisement, Tamper-resistant software, Lightweight security.

1 Introduction

Use of the Internet, especially the world-wide-web (the web), is rapidly growing. Accompanying this growth is a speedy emergence of new electronic services offered over the Internet. Many of such new services require metering in terms of the length of time or the number of instances that the services are used.

In this paper we propose an auditable metering scheme that is suitable for web advertisement applications. In our model, there are three entities involved: an advertisement content provider (we equate it to the advertiser), a service provider (e.g. Internet Service Provider (ISP)) and a client (i.e., a customer). The ISP runs a web advertising service by displaying web-based ad pages for the advertiser, and the client visits the web pages and browses ads. In general, both the ISP and the advertiser will naturally like to encourage the client to spend more time on web pages which contain ads. For this reason, the client does not have to pay for browsing ad pages. Under this model, we assume the following payment agreement between the ISP and the advertiser: the fee for an ad paid by the advertiser to the ISP is dependent on the popularity of the web page that contains the ad, i.e., the amount of time that this ad has been accessed for by the clients. It is important, therefore, for the advertiser to be able to measure

the popularity of a web page in order to determine an appropriate charge for an ad.

For the purposes of building such a web advertising service and payment model, we will propose a cryptographic scheme for metering the duration and/or the number of instances of running a process. This scheme has an attractive property: knowing a secret, one can validate a piece of metering evidence in constant units of time while without the secret the job of generating a piece of valid evidence requires time indicated by the evidence. Because the scheme utilises a well-known computational complexity problem, the meter based on it can be implemented in software yet resists tampering.

1.1 Motivation

The electronic commerce potential of the Internet, in particular, that of the web, has brought forward a new business of offering free access to the Internet. Organisations, such as Yahoo! (www.yahoo.com), Geocities (www.geocities.com), BT (www.btinternet.com), Dixons (www.freesevice.net), and TESCO (www.tesco.co.uk/indexn.htm), are a few examples of free Internet Service and/or Content Providers (ISPs/ICPs). Web advertisement is considered one important source of revenue for free ISPs/ICPs. The current figures show that advertisement on the Internet is a multi-billion dollar industry in the year 2000 [5]. Compared with the traditional hardcopy-printing-based advertisement, the web-based version is cost effective, speedy and can be conveniently connected to shopping over the Internet. For example, after viewing an ad, a client can order goods right away. Another important advantage of the web-based over the paper-based advertising is the ease of collecting data relating to clients' purchasing behaviour and of mining that information. Such information is a valuable commodity for a seller.

As a rapidly growing application, the web advertisement requires more research effort in order to identify and formulate suitable service and payment models and various related metering problems, which will in turn stimulate new research topics in computer security.

1.2 Previous Work

There are many existing commercial enterprises that try to sell services for measuring the activity of web sites (a partial list of these, in [6], includes companies like I/PRO, Nielsen, NetCount, RelevantKnowledge, and others). These companies use mainly two methods: sampling the activities of a group of web clients, and installing an audit module in web sites. Naor and Pinkas [6] argued: (i) sampling could be very inaccurate; and (ii) an audit module installed in web sites cannot prevent the web servers from forging auditing results. Thus, neither is suitable for web advertisement applications.

Franklin and Malkhi [2] initiated the work of metering via a tamper-resistant software approach (a revised version of the paper is in [3]). In their paper, Franklin and Malkhi addressed the problem of artificial client visits. They provided a lightweight security solution, which is based on a timing function that

makes a large number of artificial visits very costly. Every time a client visits a web page, the client's computer will run the timing function, and then the result of the computation will be sent to an auditing proxy. Since the timing function used in their scheme is simply repeating the construction of a hash function value and it is infeasible to invert such a hash function value (using MD-5 or SHA), to verify correctness of such evidence accurately must use the same procedure for the evidence generation. To reduce the computation of the verification, they introduced two approaches: a statistical auditing function and an approximate auditing function. Both of these functions check a part of evidence values and provide a probabilistic verification result. The drawback of their scheme is that in order to ensure that the verification is reasonably accurate, the auditing process cannot be done in constant units of time.

In Eurocrypt'98, Naor and Pinkas proposed a metering scheme [6]. In that paper, they studied methods for performing secure web metering using robust secret sharing schemes. Their approach provides computationally secure metering, which relies on the involvement of a trusted third party (an auditing agency) to compute and distribute periodically secret shares to both the web server and all potential clients. A simplification of their method is as follows: a secret is distributed by the audit agency to all N clients, and the server needs to receive requests from at least K clients in order to reconstruct the secret. The secret is then sent by the web server to the audit agency as a proof of having received at least K hits during the audit period. The main drawback of this proposal consists in the fact that clients need to be initialised for each audit period, and this forces them to communicate with the audit agency. Furthermore, this scheme cannot be used to a system, which is open to all clients who do not need a membership to use it. Clearly, this is not an open system, and therefore it is not suitable for a web advertisement application.

Bergadano and Mauro, [1], proposed a hardware-based tamper-resistance solution. They require the web server to be equipped with a special tamper-proof hardware that is provided by the certification agency. The drawback of this proposal is that the site's web server will have to be modified.

1.3 Our Work

We propose a software-based cryptographic scheme, which utilises a well-known computational complexity problem, for metering the length of time or the number of instances of a process.

We make use of this scheme to solve the metering problem in a web-based advertisement service and payment model. The main difference between this model and that in the previous work is that we will first argue that an advertiser is an important entity in the web advertising and billing services, so that we designate this entity to act as an auditor. It is however not necessary for the advertiser to be on-line during the time when a piece of metering evidence is generated. We only need the advertiser to be involved in the system initialisation stage for the generation of some primary parameters for the system. The verification of the metering evidence will be carried out in an off-line manner.

Similar to the auditable metering scheme of Franklin and Malkhi [3], our scheme can be implemented without changing any existing structure of web servers or browsers, and our solution can provide a timing measure for client visits in a lightweight security way. By lightweight security in a web advertisement application, we mean that it is computationally infeasible for anyone, except for one knowing a secret, to forge an amount of valid timing evidence without spending time that is indicated in the evidence. Without knowing a secret, the job of generating a huge amount of valid client visit evidence requires time that is indicated in the evidence. As it was argued in [3], lightweight security (which has been discussed for many different applications and the first example was in [4]) is a good strategy for metering web popularity.

Compared with Franklin and Malkhi approach, our scheme provides a more rigorous and more efficient verification procedure. The verification procedure used in our auditable metering scheme gives an accurate result instead of a probabilistic one as in their solution. Furthermore, in our scheme, any piece of metering evidence, no matter how big the amount of the time duration or the number of instances indicated in the metering evidence is, can be verified in constant units of time. The time complexity for verifying a piece of metering evidence is independent of the amount of the time that the evidence has been generated.

1.4 Organisation

The rest of this paper is organised as follows. In Section 2, we present our auditable metering scheme. In Section 3, we first identify a web-based advertisement service and payment model, secondly address some security requirements, and finally discuss implementation issues. We conclude in Section 4.

2 An Auditable Metering Scheme

We present two algorithms. The first is called Timing Algorithm, and the second, Auditing Algorithm.

2.1 The Timing Algorithm

Let $n = pq$ be an integer with two large prime factors p and q which are roughly the same size; x and e be two random integers less than n where e is co-prime to $\lambda(n)$ and hence is odd. Timing Algorithm inputs the tuple (n, x, e) and outputs (t, a) with the following computations.

Timing (n, x, e)
 $y \leftarrow h(x); a \leftarrow y; t \leftarrow 1;$
while there is no “stop” interruption {
 $a \leftarrow ya^e \pmod n;$
 $t \leftarrow t + 1;$
}

return (t, a) ;

end,

where

$h()$ denotes a collision-resistant hash function that the system has agreed;
 $\text{symbol} \leftarrow$ means “is made equal to” (for example, in $y \leftarrow h(x)$, variable “ y ” is made equal to the value of integer “ $h(x)$ ”).

Upon termination, the tuple (t, a, x, e, n) constitutes a piece of metering evidence.

Timing (n, x, e) iterates while a computer is executing a task (for instance, the client is browsing a web page). Each iteration represents one “tick”, which may in turn represents a number of fixed units of time. The number of ticks accumulated is represented by the value of t . The output pair (t, a) satisfies:

$$a \equiv h(x)^b \pmod{n}, \quad (1)$$

where

$$b \equiv 1 + e + e^2 + \cdots + e^t \pmod{\lambda(n)}. \quad (2)$$

Here $\lambda(n) = \text{lcm}(p-1, q-1)$.

Clearly, to generate a valid piece of metering evidence, Timing (n, x, e) needs to execute t iterations, each contains a modulo exponentiation. We should emphasise that without knowing the factorisation of n , t exponentiations seem to be necessary: for $n \gg t$, to date no algorithm exists that can generate a piece of valid metering evidence with time less than that of t exponentiations mod n . A similar idea for making a time measurement based on factoring problem is used in [8].

2.2 The Auditing Algorithm

Let (t, a, x, e, n) be a piece of metering evidence generated by Timing (n, x, e) . Auditing Algorithm inputs the tuple (t, a, x, e, n) and outputs YES or NO.

Auditing (t, a, x, e, n)

$y \leftarrow h(x)$;

$E \leftarrow e^{(t+1)} \pmod{\lambda(n)}$;

if $(a^{(e-1)} \equiv y^{(E-1)} \pmod{n})$ {

return (YES);

}

else{

return (NO);

}

end.

Clearly, with $\lambda(n)$, Auditing (t, a, x, e, n) will terminate in three modulo exponentiations and answer YES or NO. The time complexity is independent of t in the metering evidence.

2.3 Analysis of the Scheme

The above scheme has the following properties.

1. *For $t \ll n$ (e.g. $t \leq 2^{100}$, $n \geq 2^{1024}$), an algorithm can generate valid evidence in time less than t exponentiations will form a grand breakthrough in the area.*

Let exponentiation modulo n take one tick (representing some given units of time). Then, generating a valid pair (t, a) using Timing (n, x, e) takes t ticks.

2. *Timing (n, x, e) is intrinsically sequential.*

Clearly, every successive tick can only be performed on the result of a previous tick; there is no obvious way to parallelise the procedure using multiple processors aiming at saving time. One may compute the exponent b' first and then perform one exponentiation modulo n . However, without knowing the order of $x \bmod n$, the exponent

$$b' = 1 + e + e^2 + \cdots + e^t \quad (3)$$

is not compact, which means that the size of b' is $t|n|$ (here $|n|$ means the bit size of n in the binary representation). Therefore, the exponentiation using the large exponent b' shall still take t ticks. A disadvantage of doing so is that a huge space is required while no time is saved.

3. *Verification of the metering evidence is efficient with knowledge of the secret $\lambda(n)$.*

To verify the metering evidence (t, a, x, e, n) using Auditing (t, a, x, e, n) only takes three ticks, since computing

$$E \equiv e^{(t+1)} \pmod{\lambda(n)} \quad (4)$$

and

$$a^{(e-1)} \equiv h(x)^{(E-1)} \pmod{n} \quad (5)$$

only involves three modular exponentiations. Auditing (t, a, x, e, n) has a constant space complexity of $|n|$, since E has the same size of n . Thus, the auditor is able to check a large amount of evidence efficiently. The efficiency is obtained as a result of the auditor's knowledge of the secret $\lambda(n)$, which allows him to compute E in (4) in a compact manner.

4. *Let $a(t)$ denote the output value a of Timing (n, x, e) after t ticks. Then without factoring n , it is infeasible to compute $a(t-1)$ from $a(t)$.*

It is obvious that any one is able to compute $a(t)$ from $a(t-1)$, x , e , and n by running one "tick" in Timing (n, x, e) . However, computing in the backward direction from $a(t)$ to $a(t-1)$ involves extracting $(e-1) - th$ root of a give number mod n (see below Property 7).

5. *Optionally, the value of e can be fixed.*

In Timing (n, x, e) , e is a random number. Alternatively, using a fixed value e instead of a random number should be fine, as long as e is a positive odd number less than n .

6. *The number x can include some special information if required.*

The basic requirement on the number x is unique and fresh. For the purposes of meeting different requirements, the number x can actually include some special information.

Example 1: this number can be constructed by using more than one random numbers; respectively each random number is contributed by one entity, e.g., the server, the client or the auditor. Thus, every contributor is able to check the freshness of the metering evidence.

Example 2: this number can include some information to specify particular servers, auditors or clients if required, such as the names or other identifiers of these entities.

7. *With the secret (p, q) , a valid piece of metering evidence can be constructed in a constant length of time.*

In Auditing (t, a, x, e, n) we have shown an efficient procedure for the auditor, who knows the secret of factorisation of n , to validate a piece of metering evidence. Furthermore, he can actually construct such a piece of metering evidence without running Timing (n, x, e) . Based on $E \equiv e^{(t+1)} \pmod{\lambda(n)}$ and $a^{(e-1)} \equiv h(x)^{(E-1)} \pmod{n}$ in Auditing (t, a, x, e, n) , in order to construct the value a , the auditor needs to extract the $(e-1) - th$ root of $h(x)^{E-1} \pmod{n}$, which is a . Knowing the factorisation of n , such roots can be computed in polynomial time in the size of n .

Since the auditor is able to forge metering evidence, this auditable metering scheme is only suitable for applications where we can trust that the auditor has no incentive to forge evidence. This is the case in our web-based advertisement services to be described in the next section, where the auditor is an advertiser, who is supposed to pay for the time that his ads are browsed based on the metering evidence generated.

8. *Without factoring n , any verifier given $a(t+1)$ is able to validate $a(t)$, if he also knows that $a(t+1)$ is a correct value of evidence.*

This is obvious. Note that with this property anyone without knowing factorisation of n is able to verify evidence. For example, a meter owner issues a tuple $(t = T+1, a = f(x, T+1), x, e, n)$ and signs it as a certificate of evidence, where $a = f(x, T+1)$ can be constructed by the owner in the way described in Property 7 above. After T units of running Timing (n, x, e) , a client gets evidence $(t = T, a' = f(x, T), x, e, n)$. The certificate and evidence prove that the client has run the programme at least T units of time and that is universally verifiable.

Given consideration to both security and performance, we should choose a proper length of size of the number n , such as a similar length of a secure RSA module [7]. We recommend using n of the length of 1024 bits.

3 Web-Based Advertisement with Auditable Metering

In this section, we introduce our auditable metering scheme for web advertisement applications. We start with identifying a web advertising service and payment model where three entities are involved, and then discuss requirements on security and implementation of the metering scheme. After that, we will present the metering scheme itself.

3.1 The Model

The three entities in our model of web advertisement services are the advertiser, the Internet Service Provider (ISP), and the Client. The advertiser provides the content of an ad. The ISP runs an advertising service for this ad in a web page via the Internet. The Client obtains the information of the ad by visiting the ISP's web site.

Both the ISP and the advertiser would naturally like to encourage the client to spend more time viewing the ad. The client does not have to pay for accessing the web pages. The ISP and the advertiser have a payment agreement: the advertising fee for an ad, which is paid by the advertiser to the ISP, is proportional to the popularity of the web page that contains the ad, i.e., the amount of the time duration when this page has been accessed for by the clients.

3.2 Requirements

Our design requires the following.

1. *Lightweight security.* The scheme should provide lightweight security (see Subsection 1.3 for the meaning of the lightweight security in a web advertisement application). To ensure this, the scheme must make no sense for the advertiser to forge metering evidences. It must not be cost effective for the ISP, without colluding with a great number of clients, to create a significantly large volume of artificial client visits. We assume that there is no incentive for a group of clients to collude together with the ISP to do so. We also assume that there is no incentive for a client either to refuse creation of metering evidence or to give a piece of false evidence, for example, by modifying the code of Timing (n, x, e) running on his computer.
2. *Performance.* For this scheme to be widely deployed, both the computation and the storage requirements of the client, of the ISP and of the advertiser sides should be reasonable.
3. *Compatibility.* No any modification is needed on either the client, the ISP or the advertiser computer in order for them to use or to support this scheme. The scheme should be deployable in the commercially available web browsers.

4. *Client privacy.* No client registration is needed. This also ensures the openness of the system.
5. *Off-line advertiser.* It should not be necessary for the advertiser to be on-line during metering evidence generation. The advertiser should only be active in the time of generating system parameters and verifying metering evidence.

3.3 The Scheme

Our scheme has the following three parts.

Part 1: Parameter generation.

The advertiser generates $n = pq$, forwards it to the ISP, and keeps p and q secret. This may be done in advance of any ad downloading request or on demand. Optionally, if the scheme makes use of a fixed value e , as analysed in Property 5 of Subsection 2.3, e may be generated by the advertiser and then sent to the ISP together with n .

Part 2: Metering procedure.

1. The client “clicks” on a hyperlink to download from the ISP a web page containing an ad. This causes the browser to generate a request message, which is sent to the ISP.
2. Upon receipt of the request from the client, the ISP generates a mobile code of Timing (n, x, e) that includes n , e and x . The ISP then sends the ad web page and the mobile code to the client.
3. Upon receipt of the above, the mobile code Timing (n, x, e) starts to execute in the client’s computer. In the meantime, the web browser of the client displays the ad web page.
4. When the client leaves the web page, the mobile code stops the execution and outputs the metering result (t, a, x, e, n) , which is then sent to the ISP.
5. Upon receipt of the metering result (t, a, x, e, n) , the ISP stores it for future billing of the advertiser.

Part 3: Auditing procedure.

Later in the payment time, if the advertiser wishes to validate the metering evidence, he can do so by running Auditing (t, a, x, e, n) .

3.4 Security Analysis

Possible threats to such a metering scheme include to create false metering evidence and to refuse to provide valid metering evidence.

Based on our web advertising service and payment model described in Subsection 3.1, there are two reasonable assumptions as we have argued earlier. The first is that there is no incentive for the advertiser to forge evidence because he

is a payer. The second is that there is no incentive as well for the client to attack the advertising web page in order to bypass the mobile code of Timing (n, x, e) because he does not pay for visiting the ad web pages. Only the ISP may try to create false metering evidence with the intention to over-charge the advertiser. Our metering scheme has the following two security features, which prevent the ISP from getting any benefit by doing this.

1. *This scheme provides a lightweight security solution, which is ensured by using the auditable metering scheme described in Section 2.* According to Properties 1 and 2 of the metering scheme as described in Subsection 2.3, the only way for the ISP to generate a piece of metering evidence is to behave as a client, which is very labour-intensive if a significant amount of evidence needs to be generated. Indeed, the time and resources taken to generate the evidence would cost the ISP more than what would be gained from the increased advertising revenue.

2. *Because the metering scheme used in this scheme utilises a well-known computational complexity problem (difficulty of factorisation), the scheme can be implemented in pure software yet resists tampering.* The ISP may also try to modify the code that computes Timing (n, x, e) on the client's computer in order to count "faster than it normally should". We first argue that the code should be well designed to compute the algorithm in the fastest way. If there is a method to implement Timing (n, x, e) faster than other methods, this method should be used to implement the algorithm. Secondly, if it is the case that the code does not implement the algorithm in the fastest way for some reason, we propose an alternative solution, where the code could be issued and certified by a trusted authority (either the advertiser or another trusted third party), and its certificate could be verified by the client's computer. If the client's computer finds the code has been modified, it would refuse to send the evidence to the ISP.

4 Conclusions

We have proposed a new auditable metering scheme based on a well-known computational complexity problem, and have also discussed how to use this scheme to implement a pure software-based temper-resistant metering scheme, which can be used in web advertising services and other Internet services. Compared with the previous metering approaches with similar designing requirements, our scheme offers a more rigorous and efficient procedure for the evidence verification.

References

1. Bergadano F., De Mauro P.: Third party certification of HTTP service access statistics. The Proceedings of the 1998 Cambridge International Workshop on Security Protocols, Cambridge, England, 1998

2. Franklin M. K., Malkhi D.: Auditable metering with lightweight security. The Proceedings of Financial Cryptography '97, LNCS 1318, Springer, Berlin, (1997) 151–160
3. Franklin M. K., Malkhi D.: Auditable metering with lightweight security. Journal of Computer Security **6**(4) (1998) 237-255
4. Glassman S., Manasse M., Abadi M., Gauthier P., Sobalvarro P.: The Millicent protocol for inexpensive electronic commerce. The Proceedings of the 4th International World Wide Web Conference (1995) 603-618
5. Kinsman M.: Web advertising 1997: market analysis and forecast. Cowles/Simba Information. Stamford, Connecticut, May 1997
6. Naor M., Pinkas B.: Secure and efficient metering. The proceedings of EUROCRYPT '98, LNCS 1403, Springer, Berlin, (1998) 576-590
7. Rivest R.L., Shamir A., Adleman L.M.: A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM **21**(2) 1978 120-126
8. Rivest R.L., Shamir A., Wagner D.A.: Time-lock puzzles and timed-release crypto. LCS Technical Memo MIT/LCS/TR-684, February 1996


Broker-Based Secure Negotiation of Intellectual Property Rights

Jaime Delgado¹, Isabel Gallego², and Xavier Perramon¹

¹ Universitat Pompeu Fabra (UPF), Departament de Tecnologia,
Pg. Circumval·lació 8, E-08003 Barcelona, Spain
{jaime.delgado, xavier.perramon}@tecn.upf.es

² Universitat Politècnica de Catalunya, Departament d'Arquitectura de Computadors,
Campus Nord, Mòdul D6, Jordi Girona 1-3, E-08034 Barcelona, Spain
isabel@ac.upc.es

Abstract. In the area of e-commerce of multimedia content, protection and management of Intellectual Property Rights (IPR) is a key issue. Using a broker as intermediary between copyrighted material purchasers and content providers helps, apart from increasing the market perspectives, to improve the way of handling the IPR aspects. The conditions in which multimedia content is sold may vary from one case to another, especially in a B2B environment. For this reason, negotiation of copyright conditions at the purchase moment is to become more and more important.

We are developing an application that represents IPR information using metadata and uses a specific protocol for negotiation purposes. All of this in a broker-based multimedia content e-commerce system. We are also experimenting with the use of agent technology to implement the negotiation protocol. The result of the negotiation is an electronic contract, that should be signed by the interested parties. Apart from specifying this data structure, we are working in the integration of an electronic signature mechanism using the XML Signature specification. 

1 IPR Business Models

During the last years, a lot of work has been done in the area of Intellectual Property Rights (IPR) management, that includes copyright protection and Electronic Copyright Management Systems (ECMS). Currently, the work in this area is even increasing, because there is a real and urgent need to solve the problems of multimedia content copyright control that the spreading of the Internet and the WWW has created. There are different European projects related to IPR management like IMPRIMATUR [1], INDECS [2], COPEARMS [3], FILIGRANE [4], etc.

Although many technical solutions to the problems already exist, there is still a lack of an efficient and agreed business model for e-commerce of multimedia content. Several solutions are being deployed, mainly in the area of music distribution, such as Soundwrap [5], DMDsecure [6], or DX3 [7], but none of them has succeeded as a

¹ This work has been partly supported by the Spanish government (TEL98-0699-C02-01 and TIC2000-0317-P4-05).

model to follow. In the area of e-books, as another example more in the e-publishing world, there are different initiatives to digital rights management, such as EBX [8] and OEBF [9], and some XML based ones, such as XrML [10].

The starting point, from an IPR's point of view, is the selection of the model in which to base IPR representation and negotiation. The IMPRIMATUR Business Model [1], the one we have selected, identifies a series of entities that may take different roles, such as Creator, Provider, Rights Holder, Distributor, IPR Data Base, or Watermarking & Fingerprint marking.

Figure 1 illustrates the different entities participating in the general model for IPR handling. It must be noted that one user may take more than one different role.

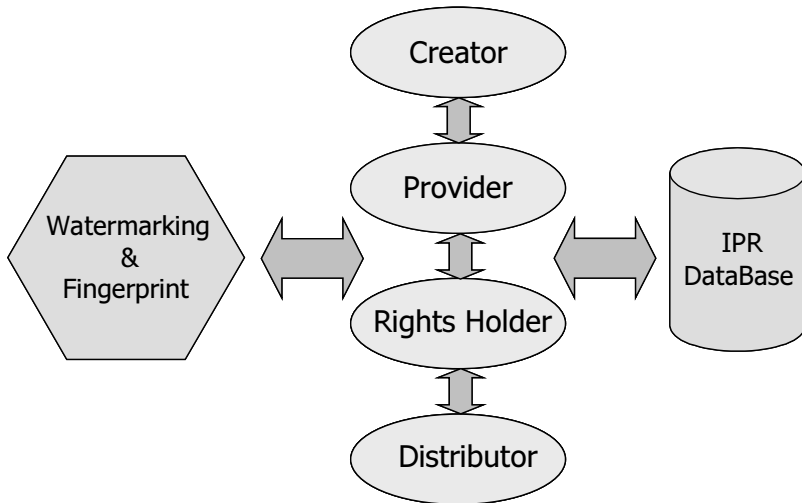


Fig. 1. Different entities in an IPR General Model

A more simplified and specific model, the one we are implementing, consists on the use of a Broker (with the role of Distributor) in charge of being an intermediary between providers of multimedia material (content providers) and customers interested in buying that material and the corresponding rights for use and/or commercial exploitation. From a functional point of view, these copyrighted multimedia material providers may also assume the roles of Creator and Rights Holder in the same entity.

Furthermore, the broker stores and keeps up to date (with the help of content providers) the information about the multimedia material for sale in the system (from all content providers associated to the broker), and about the terms and conditions in which commercial electronic transactions are done, with the help of the IPR Database. Figure 2 illustrates this Broker Based IPR General Model.

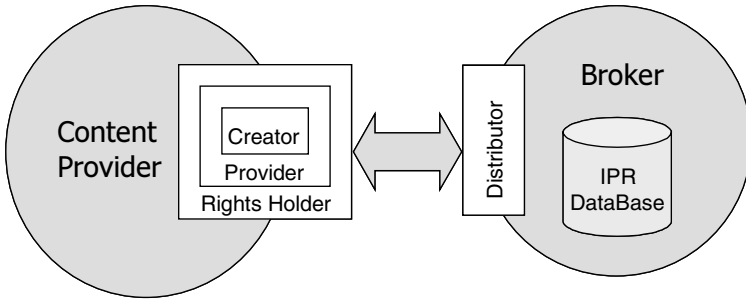


Fig. 2. Broker Based IPR General Model

2 IPR Information Representation

For managing IPR information, we need to represent it in some way. We have chosen a solution based on the use of metadata. A lot of work is going on in the area of metadata, from the widely used Dublin Core [11] until the MPEG7 standard [12] for audiovisual material. Our work is based on the alignment efforts we started in the context of an European CEN/ISSS Workshop [13], and on the work going on in the INDECS project [2], that focuses on metadata for IPR. We are currently implementing this solution in different projects, from brokerage of audiovisual material (as we did in the HYPERMEDIA Project [14]), to broadband multimedia applications (the MARS Project [15]).

Basically, we use two kinds of metadata: general and IPR. General metadata, on the one hand, includes management (title, author(s), kind of material, size, etc.) and technical information (resolution, colour, shape, etc.). On the other hand, IPR metadata includes attributes giving information associated to watermarking, rights owner, kind of rights, rights status, temporal and spatial restrictions on rights, etc.

This model just introduced has been implemented with RDF [16], extending the fundamental modelling concepts provided by the “RDF model and syntax” [17] and the “RDF schema” [18]. The schema utilisation policy of this framework allows the creation of a very flexible representation, untied to any concrete initiative of metadata creation.

To start developing from a solid basis, the model has been constructed over INDECS basic model. This provides the general concepts that allow non-traumatic extensibility and a great adaptability to concrete and changing environments.

The descriptive metadata subset includes all those audiovisual material properties that permit the easy and rapid localisation of the audiovisual material needed. These properties can be packaged inside RDF schemes. A more detailed description of our metadata approach is given in [19].

3 Negotiation of IPR Conditions

Based on the IPR attributes set, when a buyer requests, to the broker, a purchase of audiovisual material subject to copyright, the broker extracts IPR information from its database (Clauses Templates) and presents an initial offer to the buyer (Contract by default). This information allows the buyer to take a decision on how to buy IPR, i.e., to know what are the copyright rules associated to the asset, to decide if to re-sell it, etc. To facilitate this process, a negotiation mechanism is being developed based on a simple negotiation protocol. XML [20] is used as interface to the IPR information in the broker's database. Since all metadata attributes are specified using XML, the implementation could be easily ported from one system to another. If the negotiation process finishes with agreement, it is complemented by the production of an electronic contract, which is signed by buyer, rights owner and broker (as TTP), and stored in the broker agent, as detailed later in section 5.

The negotiation protocol, that it is part of the "Service Request" phase in an e-commerce model [21], has three sub-phases:

- Initial offer (Contract by default),
- Co-operative contract production (kernel sub-phase) and
- Payment.

The initial offer (Contract by default) is selected, between a set of clauses templates stored in the database of the broker agent, with the `Default_Contract` operation:

```
ElectronicContract    Default_Contract {
    PartiesList        negotiationList;
    ContractClauses    contractClauses; }
```

In the Contract production (or kernel) sub-phase, the most complex and important one, there are several alternatives over which to work. First, the selling entity initiates the protocol with an initial proposal of digital rights conditions, normally taken from a pre-defined subset (Clauses Templates). After that, the buying entity has three alternatives:

1. Accepting the offer (`Accept_Clause` operation),
2. Making a counter- proposal (`Add_Clause` operation) and
3. Rejecting the proposal (`Remove_Clause` operation).

After the initial proposal, the negotiating entities elaborate the contract, using the negotiation protocol, from the sequence of proposals and counter-proposals until a final agreement is reached, forming then the final electronic contract. This implies the different operations presented in the following.

The graphic representation of the kernel sub-phase of this IPR negotiation protocol is that of Fig. 3. More technical details are given in [22].

Accepting the offer is to accept a proposed clause of a contract (to add this entity to the list of entities who agree with this clause) with the `Accept_Clause` operation:

```
ElectronicContract    Accept_Clause {
    Name               agreeEntity;
    int                identifierContractClause;
    int                contractSerialNumber; }
```

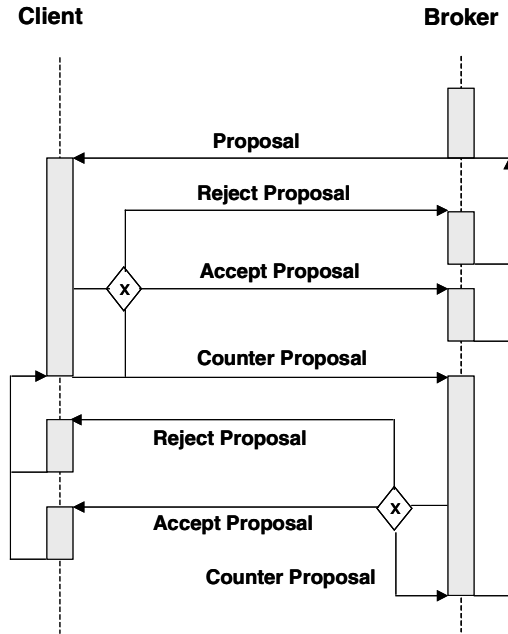


Fig. 3. Kernel sub-phase of the IPR Negotiation Protocol

Making a counter-proposal is to add a new clause to the contract with the `Add_Clause` operation:

```

ElectronicContract  Add_Clause {
    int    identifierContractClause;
    int    contractSerialNumber; }
    
```

Rejecting the proposal is to remove a clause of the contract with the `Remove_Clause` operation:

```

ElectronicContract  Remove_Clause {
    int    identifierContractClause;
    int    contractSerialNumber; }
    
```

This “game” of proposals and counter-proposals could be extended under some rules (Negotiation Protocol). The negotiation is done through the broker, who has the unique legal copy of the contract. When the contract negotiation is finished, the broker generates the legal copy of the contract (Signed contract, see section 5) and sends a copy of the contract to the involved entities.

More operations are necessary related to:

- Security: Sign, Authentication, Integrity, Hashing, Check_Signature, Check_Integrity.

- Digital Marks: Put_Mark, Get_Mark, Check_Mark, Add_Mark, Remove_Mark.
- Payment: Select_Payment, Set_Bill.
- List Management: Join, Remove.

Finally, although the rights negotiation protocol concludes when a proposal is eventually accepted or rejected, the IPR-related protocols continue in the context of the e-commerce “Agreement” phase [21], since a “Post-negotiation” step should be added, where the contract is executed and the fulfilment of the agreements in the contract is verified, following some workflow rules. We are already implementing a negotiation protocol with agent technology [23]. However, it is in this “Post-negotiation” phase where this agent technology plays a more important role (see [19]).

4 IPR Contracts

Initially, the creator (writer, composer, actor, software programmer, etc.) of a work is holder of all the rights over it. Authors may not have the economic power to exploit their creations in an effective way. In such cases some or all exploitation rights can be transferred to publishers/producers specialized on this task.

This can be formalised in *author's contracts*. By them, a publisher/producer becomes the exploitation rights holder of the work (Rights Holder). As a counterpart, the originator will receive royalties for the benefits derived from the exploitation of the work, or/and an initial fix revenue.

Publishers/producers do not usually have direct contact with end-users. They commercialise works through distributors (bookshops, libraries, Internet site providers, broadcasting companies, etc.), entities specialized in acquiring part of the rights of copyrighted works to allow them to satisfy final users demand of these works. Distributors may also encapsulate works with some services that facilitate this consumption, the added value they provide to the creations. Formalization of these agreements is made through *distribution contracts*, that may transfer some constrained exploitation rights or be mere licenses allowing the required actions to complete the commercialisation to end-users.

Finally, through distributors, copyrighted works arrive to their consumers. End-users acquire the right to access and use them by *purchase contracts*. These use rights may operate within the framework established in the distribution contract with the publisher/producer. The income a distributor receives derived from these purchases will be also distributed, in a cascade manner according to the royalties established in the different contracts so far, to publishers/producers and originators implied in the elaboration of the final product acquired by the consumer.

To ensure that fair economic returns arrive to authors and publishers/producers, they can establish membership forms with collecting societies. They control the exercise of representation and reproduction rights of some types of intellectual works to enforce proper remuneration to their members.

4.1 An Electronic IPR Contract

Independently from the kind of agreement reached by means of the negotiation protocol already described in section 3, once it is reached, there is a need to produce a contract to formalise the agreement. In the next clauses, we describe how to specify an electronic contract for this formalisation.

As already mentioned, by IPR negotiation we mean the process in which, at purchase time, the buyer of some multimedia content and the rights owner (or representative) negotiate the conditions (concerning rights) in which that material is sold. This process, run through a protocol with some interchange of information, is equivalent to creating an electronic contract. It could be also seen as a joint editing of a structured document (the contract), following pre-specified alternative rules.

The electronic contract, that should be electronically signed, has two parts:

- Mandatory part: It contains the minimum information necessary to formalise an electronic contract.
- Optional part: It contains optional information related to any kind of contract.

```
struct ElectronicContract {
    MandatoryPart    mandatoryPart;
    OptionalPart      optionalPart; }
```

The mandatory part includes the following information:

- Type: To identify the kind of contract.
- ContractSerialNumber: Contract unique identifier
- SignatureInformation: Information related to the signature of the contract.
- NegotiationList: List of entities who participate in the negotiation of the contract.
- Broker: Identification of the broker who has the TTP role.
- Validity: Validity range of the contract.

```
struct MandatoryPart {
    Type          type;
    int            contractSerialNumber;
    Signature      signatureInformation;
    PartiesList    negotiationList;
    Broker         broker;
    Validity       validity; }
```

The optional part is a sequence of contract clauses.

```
struct OptionalPart {
    ContractClauses    contractClauses<>; }
```

Every contract clause includes the following information:

- IdentifierContractClause: Clause unique identification.
- Material: Multimedia material affected by this clause.
- Right: Kind of right affected by this clause (right transfer, etc.).
- AgreeList: List of entities who agree with this clause.

- Obligations&FallsList: List of obligations and associated punish actions.
- PaymentInformation: Payment information involved with this clause.
- ClauseDescription: Text description of the clause.

```
Struct    ContractClause {
    int          identifierContractClause;
    Material     material;
    TypeRight    right;
    PartiesList  agreeList;
    Obligation&Fall obligations&FallsList<>;
    Payment      paymentInformation;
    string       clauseDescription<>; }
```

5 Electronic Contract Signature

In our IPR management application, all users, either content providers or buyers, must agree and explicitly accept that contracts are binding and enforceable once they are signed by the parties involved. In these electronic contracts, the equivalent to traditional hand-written signatures is achieved with public key cryptography techniques. Every user of the system has a pair of keys (private/public) for generating signatures. The broker also has a key, trusted by all users, that is used to certify that keys actually belong to their owners.

We have decided to base our implementation of contract signatures on the emerging XML Signature standard, developed by the W3C and the IETF [24, 25]. This choice was made for the following reasons:

1. The syntax used in XML Signature, as its name implies, is an application of the XML language. Therefore, XML processing tools, such as parsers, editors, converters, style processors, formatters, viewers, etc. can be used for handling the syntactical structure of signatures.
2. One of the strengths of XML is the availability of tools in many diverse platforms, which facilitates interoperability between implementations.
3. The XML Signature technique is designed to be applicable to any resource, i.e. any object addressable through a URI. In particular, HTML and XML documents are natural candidates to be signed with XML Signature, and our contract representation is based on XML.
4. Another instance of resources that can be signed with XML Signature are XML document fragments. This allows for the possibility, to be considered in the future, of establishing generic contracts and having every party sign only the applicable clauses in each case.
5. Since an XML Signature is represented as an XML document element, it can be incorporated into any XML document. In the case of valid XML documents (as opposed to simply "well-formed" documents, according to the XML terminology), this may require amendment of the DTD or Schema used in the

document. In our contract implementation, however, the document type is already designed to provide for the inclusion of signatures.

6. The data model for specifying an XML Signature is based on RDF (although its representation does not necessarily have to use the RDF syntax). This makes it simpler to integrate signatures in an environment where metadata are used to describe data, as it is the case in our IPR model.
7. According to the XML Signature Requirements specification [26], the needs of several already existing electronic commerce oriented applications, such as IOTP (Internet Open Trading Protocol) [27], should be fulfilled with XML Signature. Our contracts application also fits in this context.

Furthermore, XML Signature offers other convenient features which are common to most digital signature applications: the possibility of having different signatures applied to the same document (this is obviously useful in the case of contracts), the capability to sign already existing (and possibly read-only) documents generating detached signatures, and independence of the actual cryptographic algorithms used in the signature.

Two normative references constitute the XML Signature specification: Canonical XML [24], and XML Signature Syntax and Processing [25]. Canonical XML defines which of the logically equivalent representations of an XML document is to be used as the input to the cryptographic algorithms. This definition is necessary because XML allows for a certain degree of variation in the physical representation of a document (i.e. the actual byte sequence) without changing its semantical meaning.

Some aspects of the physical representation, such as attribute ordering or replacement of references by their values, are irrelevant for most XML applications. However, the Canonical XML specification identifies potential limitations whereby meaningful information could be lost as a result of canonicalization. For example, the document type definition (DTD) can be used for deriving default values for omitted attributes, but after it has been processed it has to be removed from the canonical representation. Certain attributes might have properties that can only be specified in the DTD, and therefore further processing applied to the canonical form might not be equivalent to processing the original document. Nevertheless, typical applications do not make use of these conflictive features, and this is the case also for our contract definition.

The second specification, XML Signature Syntax and Processing, defines the XML elements and attributes that shall be used for the representation of signatures, and the rules for their generation and their validation. The XML elements conveying the signature information can be “enveloping” if the signed data is a sub-element of the signature, “enveloped” if the signature is a sub-element of the signed data, or “detached” if the signature and the signed-data are in disjoint sub-trees of the same document or in separate documents.

The preferred signature representation in our contract implementation is through enveloped elements: detached signatures do not provide any advantage, since signatures are an essential part of the contract (a contract is of no value without the signatures), and enveloped elements are preferred over enveloping elements because the former facilitate the association of several signatures with the same data.

The public key material necessary for validating an XML Signature may be included within the signature itself by value, or referenced through URIs. The URI

method is appropriate when using certificate chains, but in our case all signing keys are certified by one single authority, which is the broker. We use X.509 [28] for representing certificates, and therefore all contract signatures include the X.509 certificate of the signer issued by the broker. The broker certificate is not included because it is a root certificate (auto-signed) and the broker public key is made known to all users when they first register into the system.

6 Conclusions

In this paper we have presented a model for the management of IPR and a system for negotiating IPR contracts based on this model. The system is oriented to the electronic commerce of multimedia content, and it is based on a broker that acts as an intermediary between content providers and buyers. The broker is in charge of taking all steps necessary for the negotiation of the conditions regarding the purchase of a work. After the parties have agreed on the conditions and the negotiation phase is completed, a XML-based contract is produced which is digitally signed by all parties involved. The technique used for representing and signing the contract is based on the XML Signature standard.

The use of XML Signatures offers significant advantages to our IPR model, given the high degree of interoperability provided by XML, its availability in many already existing browsers, and its easy integration into metadata-based electronic commerce applications.

The system we have presented here facilitates the implementation of electronic commerce applications for the distribution of copyright protected electronic material.

References

1. IMPRIMATUR Project, <http://www.imprimatur.net/>
2. INDECS Project, <http://www.indecs.org/>
3. COPEARMS, Project, <http://www.ispo.cec-be/istka2/c4/actionline43.htm>
4. FILIGRANE Project, <http://www.filigrane.org/>
5. Soundwrap (Virtual Shrink-wrap for Music Distribution), <http://www.soundwrap.com/>
6. DMDsecure (Digital Media Distribution), <http://www.dmdsecure.com/>
7. DX3 (Digital Distribution Domain), <http://www.dx3.net/>
8. EBX (Electronic Book eXchange), <http://www.ebxwg.org/>
9. OEBF (Open eBook Forum), <http://www.openebook.org/>
10. XrML (eXtensible rights Markup Language), <http://www.xml.org/>
11. DUBLIN CORE Home Page, <http://dublincore.org>
12. MPEG-7, <http://www.cselt.it/mpeg>
13. Metadata for Multimedia Information Workshop, CEN Information Society Standardisation System, <http://www.cenorm.be/issw/Workshop/delivered-ws/MMI/Default.htm>
14. HYPERMEDIA Project, <http://www.corporacionmultimedia.es/hypermedia>
15. MARS (Multimedia Advanced brokerage and Redistribution Surveillance) Project, <http://www.upf.es/esup/dmag>
16. RDF (Resource Description Framework), <http://www.w3.org/RDF>
17. RDFMS (RDF Model and Syntax Specification), <http://www.w3.org/TR/REC-rdf-syntax>

18. DFS (RDF Schema Specification), <http://www.w3.org/TR/rdf-schema>
19. Gallego, I., Delgado, J., García, R.: "Use of Mobile Agents for IPR Management and Negotiation", 2nd International Workshop on Mobile Agents for Telecommunication Applications (MATA 2000), 18-20 September 2000, Paris (France). Springer, ISBN 3-540-41069-4.
20. XML (eXtensible Markup Language), <http://www.w3.org/XML>
21. Gallego, I., Delgado, J., Acebrón, J.J.: "Distributed Models for Brokerage on Electronic Commerce", International IFIP Working Conference Trends in Electronic Commerce (TREC'98), 3-5 June 1998, Hamburg (Germany). Springer, ISBN 3-540-64564-0.
22. Delgado, J., Gallego, I.: "Distributed Negotiation of Digital Rights", International Conference on Media Futures 2001, 8-9 May 2001, Florence (Italy).
23. FIPA (Foundation for Intelligent Physical Agents), <http://www.fipa.org/>
24. Canonical XML, <http://www.w3.org/TR/2001/REC-xml-c14n-20010315> (also published as RFC 3076).
25. XML Signature Syntax and Processing, <http://www.w3.org/TR/2001/CR-xmldsig-core-20010419/>
26. XML Signature Requirements, <http://www.w3.org/TR/1999/WD-xmldsig-requirements-19991014> (also published as RFC 2807).
27. Internet Open Trading Protocol (IOTP) Version 1.0, <http://www.ietf.org/rfc/rfc2801.txt>
28. ITU-T Recommendation X.509, Information technology - Open Systems Interconnection - The directory: Public-key and attribute certificate frameworks.

Design of the Decision Support System for Network Security Management to Secure Enterprise Network

Jae Seung Lee¹ and Sang Choon Kim²

¹ Information Security Technology Division, ETRI
161 Kajong-Dong, Yusong-Gu, Taejon, 305-350, Korea
jasonlee@etri.re.kr

² Dep't of Information & Communication Engineering/ Professor,
Samchok National University
253 Gyodong, Samchok, Kangwondo, 245-711, Korea
kimsc@samchok.ac.kr

Abstract. Decision Support System for Network Security Management is a system that evaluates the security of a network domain consists of various components and that supports a security manager in making decisions about security management of the network based on the evaluation. It helps the security manager to make a decision about how to change the configuration of the network to prevent the attack due to the security vulnerabilities of the network. Decision Support System for Network Security Management checks the current status of the network, predicts the possible intrusion and supports decision-making about security management to prevent the intrusion in advance. In this paper we analyze the requirements of the Decision Support System for Network Security Management that automates the security evaluation of the network and that supports decision-making about security management to secure the network, and we propose a design for it that satisfies the requirements. We also provide a prototype that implements the basic functions of our design.

1 Introduction

1.1 Introduction of the Decision Support System for Network Security Management

Recent explosive increase in the use of the network makes the security of the network much more important than that of hosts. In addition to the Firewall [12] and the Intrusion Detection System [11], we need a tool that can check the current security status of the network. To maintain the security of a network, the tool must have the capability of evaluating the security of the network and the capability of decision support for the network security management to prevent the intrusion caused by the security vulnerabilities of the network.

Security Evaluation requires various kinds of test and analysis, and it is hard for a network manager to perform all the tests and analyses for the large network for him-

self. Therefore, a tool is needed which automates the tests and analyses. The tool must have the capability of the security evaluation of the network domain as well as the capability of the security analysis of the individual host or subnetwork. It must support a security manager in making decisions about security management of the network based on the security evaluation result to prevent intrusion caused by the vulnerability of the network.

Decision Support System for Network Security Management (DSS/NSM) is a system that evaluates the security of a network domain using automated analysis tools and that supports decision-making about security management based on the evaluation to prevent the intrusion in advance. It helps security managers to make a decision about how to change the configuration of the network to prevent the attack due to the security vulnerabilities of the network. The Intrusion Detection System [11] detects an attack when the intrusion happens but the DSS/NSM checks the current status of the network, predicts the possible intrusion and prevents the intrusion in advance.

1.2 Necessity of the Research on the DSS/ NSM

So far there is no formal and standardized technology for DSS/NSM, and vulnerability assessment tools developed by individual software companies are used for security evaluation of the network instead.

Hacking tools such as sscan, SATAN [8], SAINT, ISS are widely used for vulnerability assessment but the individual tools are not integrated and it is hard to get integrated security evaluation report from the vulnerabilities found in each tool. The tools have to be updated whenever new vulnerability is found and new version of the tool is available reflecting the vulnerability. They just provide the report on several well-known vulnerabilities from the network scanning.

Commercial vulnerability assessment tools such as AXENT NetRecon [3], ISS Internet Scanner, System Scanner, Database Scanner [4] and RSA Kane Security Analyst [5] are used for security evaluation but these tools just provide several host or network scanning functionality.

Security Evaluation of the systems that provide security services, such as VPN servers and Firewalls, is much more important than that of other systems, because the failure of these systems makes the entire network domain insecure. But there is no tool that checks that these important systems work properly and provide their security services.

DSS/NSM has to be evolved to provide automatic update of the search capability for the new vulnerability check, vulnerability analysis using hacking simulation, and analysis of the possible intrusion due to the specific vulnerability. It also has to provide integration with the Security Policy Server, vulnerability recovery functionality integrated with security management systems, decision support for the security management and scalability for large networks. But so far the research on the DSS/NSM is not enough, and there is no tool that supports all the functions described above. Therefore, further research and development are needed for the progress of the DSS/NSM Technology.

In this paper, we analyze the requirements of the DSS/NSM that automates the security evaluation of the network and that supports decision-making about security management to prevent the intrusion, and we propose a design for it that satisfies the requirements. We also provide a prototype that implements the basic functions of our design.

2 Related Works

2.1 Network-Based Scanners and Host-Based Scanners

Network-based scanners and host-based scanners [1] provide basic functions for the security evaluation. A network-based scanner performs quick, detailed analyses of an enterprise's critical network and system infrastructure from the perspective of an external or internal intruder trying to use the network to break into systems. It analyzes network-based devices on the network and quickly provides repair reports to allow quick corrective action. It can be set up and used quickly because it requires no host software to be installed on the system being scanned.

Host-based scanning's strengths lie in direct access to low-level details of a host's operating system, specific services, and configuration details. A host-based scanner can view a system from the security perspective of a user who has a local account on the system [1, 2]. It detects signs that an intruder has already infiltrated a system. It also detects any unauthorized changes in critical system files.

2.2 Commercial Vulnerability Assessment Tools

AXENT NetRecon [3], ISS Internet Scanner, System Scanner, Database Scanner [4] and RSA Kane Security Analyst [5] are famous vulnerability assessment tools in use. They provide network-based and host-based scanning functionality. They also provide Graphic User Interface and some product can analyze the communication devices such as router. Some product suggests corrective actions when the vulnerability is found.

3 Design of the DSS/ NSM

In this section, we analyze the requirements of DSS/NSM that automates the security evaluation of the network, and that supports decision-making about security management of it to prevent the intrusion caused by the vulnerability of the network. Then we propose a design for DSS/NSM that satisfies the requirements.

3.1 Requirements of the DSS/NSM

The requirements of the DSS/NSM are as follows.

- It must support Security Evaluation of the network consists of various components and it must be scalable.
- It must be able to analyze the vulnerability of a host. For example, it must be able to check system configuration error, Trojan horses and system file integrity. It also must trace signs of the hacker's intrusion.
- It must perform vulnerability checks and security analyses of each subnet. For example, it must be able to analyze the vulnerability of network services.
- It must provide generation of security evaluation reports. It also must support a security manager in making decisions about security management of the network based on security evaluation results to prevent intrusion to the network.
- It must analyze whether the various components of the network obey the security policy or not using the Security Policy Server.
- It must support remote and central administration. It must provide the central view of the security status of the entire network.
- It must provide Security Evaluation of the specific systems that provide security services to the network domain such as VPN servers and Firewalls.
- It must provide vulnerability recovery function with security management systems.
- It must get the new vulnerability information from the directory service frequently and apply it to the security evaluation.

3.2 Architecture of the DSS/NSM

Fig. 1 illustrates the architecture of the DSS/NSM that satisfies the requirements described above. The major components of DSS/NSM are Agents, Subnet Analyzers, a Domain Analyzer, a Security Evaluation Rule Manager and a Manager Tool.

3.2.1 Agent

Agents are executed on each host and they analyze the security of the host in detail. They access the low-level detail of a host's operating system, specific services, and configuration details. Each Agent analyzes the system configuration errors, signs of the hacker's intrusion, system file integrity and the existence of the Trojan horses or viruses. Each Agent then generates the security evaluation report of the host and sends it to the Subnet Analyzer that is in the same subnet.

3.2.2 Subnet Analyzer

Subnet Analyzers evaluate the security of each subnet from the viewpoint of network user. Each Subnet Analyzer analyzes vulnerability of the network services and daemons. It analyzes the host security evaluation reports from each Agent in the subnet and finds out the vulnerability that can be only checked with the information of several hosts that are part of the distributed environments. The subnet security evaluation report is sent to the Domain Analyzer.

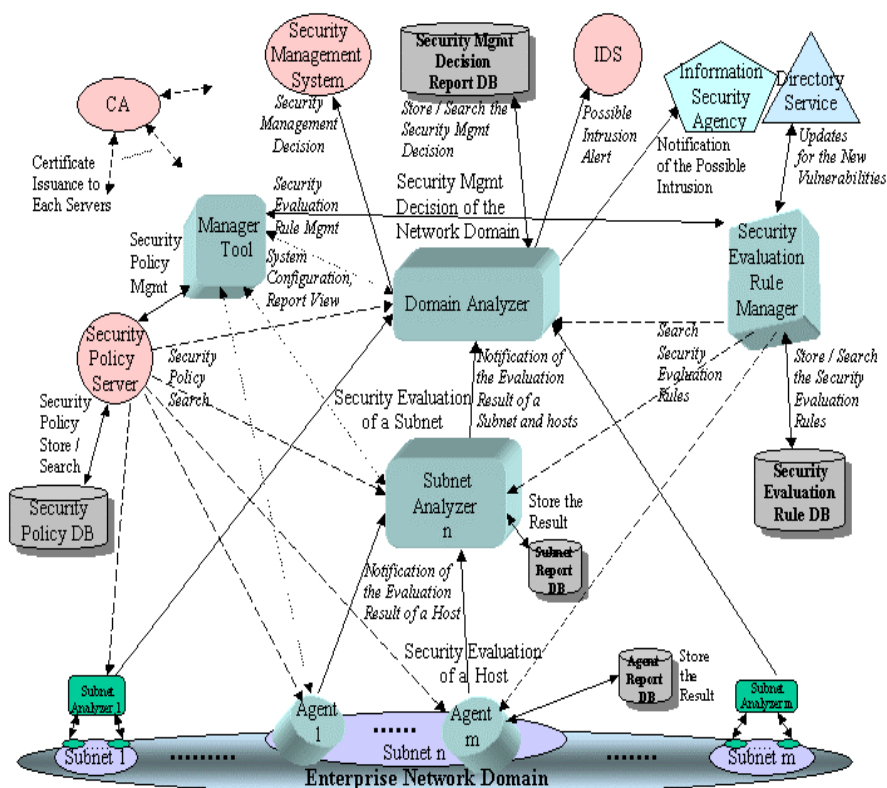


Fig. 1. Architecture of the DSS/NSM

3.2.3 Domain Analyzer

Domain Analyzer evaluates the security of the entire network domain based on the security evaluation results from each Agents and Subnet Analyzers. It analyzes the reports from each analyzer and finds out correlation between the vulnerability. It predicts the possible intrusion due to the vulnerability analyzed so far. Then it makes decisions about security management to prevent the possible intrusion and report them to the security manager. It also sends the notification mail to the system manager of the host that has the security weaknesses. It notifies the decisions about security management to the Security Management System or to the Intrusion Detection System [11] and helps them to prevent the possible intrusion.

3.2.4 Security Evaluation Rule Manager

Security Evaluation Rule Manager helps the security manager to generate and manage the evaluation rules that are needed for security evaluation. Security Evaluation rules consist of vulnerability information related to the specific host or network resource, possible intrusion due to the vulnerability, and countermeasures for it. Security

Evaluation of Agents, Subnet Analyzers, and Domain Analyzer is based on the security evaluation rules managed by the Rule Manager. Rule Manager gets the new vulnerability information from the external trusted directory service frequently and updates the security evaluation rule based on it. Therefore DSS/NSM can cope with the new vulnerability rapidly.

3.2.5 Manager Tool

Manager Tool helps the administration of DSS/NSM. A security manager can configure each component of DSS/NSM remotely. He can centrally manage the security evaluation rules and security policy. Manager Tool helps the security manager to view the evaluation reports in various formats.

3.2.6 Etc

Each component of the DSS/NSM such as Agents, Subnet Analyzers, Domain Analyzer, Manager Tool and Security Evaluation Rule Manager encrypts the data that are confidential such as security evaluation reports. For the authentication between the components, they use the certificates issued by the trusted CA (Certificate Authority).

Security Policy Server is the server that manages the security policy of the network domain. DSS/NSM always references the security policy when it performs the security evaluation and checks whether the network resource obeys the security policy or not. It makes the components of the network follow the consistent security policy.

3.2.7 Features of the System Design

DSS/NSM proposed in this paper evaluates the current security status of a network domain and supports decision-making about security management to remove the vulnerability of the network and to prevent the possible intrusion. Security Evaluation performed by each analyzer module (Agent, Subnet Analyzer, Domain Analyzer) is based on the security evaluation rules that are managed by the Rule Manager. Rule Manager gets the new vulnerability information from the external trusted directory service frequently and updates the security evaluation rule based on it. DSS/NSM can cope with the new vulnerability rapidly by this feature.

In this design, Security analysis processes are distributed over several analyzers. That is, an Agent analyzes the security of a host, a Subnet Analyzer analyzes the security of a subnet and Domain Analyzer analyzes the results from all analyzers. Therefore the processing load of each analyzer is relatively small. If the number of hosts in the network domain increases, overall evaluation performance will be decreased. But the overall performance can be preserved from decrease by adding new Agents and Subnet Analyzers to the network domain, since the processing load can be distributed over new analyzers. That is, DSS/NSM is scalable. DSS/NSM requires relatively small network traffic because not all the host or network information is sent to the Domain Analyzer. Agents or Subnet Analyzers gather all the information about hosts or subnet, analyzes the hosts or subnet information, and they send the security evaluation report to the Domain Analyzer. The report describes only information related to the security evaluation of the host or the subnet. It also greatly reduces the amount of

data that the Domain Analyzer has to analyze. Therefore, DSS/NSM is suitable for large network.

Each analyzer module in DSS/NSM references the security policy of the network domain and checks whether the components of the network obey the security policy whenever the analyzer performs the security evaluation of the component. It makes the components of the network follow the consistent security policy.

3.3 Detailed Architecture and Processing Flow of DSS/NSM

An outline of the processing flow of DSS/NSM is as follows. Update of the Security Evaluation Rule DB is performed frequently independent of each analyzer in DSS/NSM.

[Step 1] Security Evaluation of each host by Agents

[Step 2] Security Evaluation of each subnet by Subnet Analyzers

*[Step 3] Decision-making about Security Management of the network
by Domain Analyzer*

An outline of the processing flow of each analyzer module (Agent, Subnet Analyzer, Domain Analyzer) is as follows.

*[Step 1] Analysis of the host or network resources based on
Security Evaluation Rules and Security Policy*

[Step 2] Generation of the Security Evaluation Result Report

*[Step 3] Notification of the Evaluation Result to the higher analyzer
module*

Both the security evaluation rule and the security policy have to be referenced during security evaluation. To do this, the analyzer selects the rules and policy that are suitable to the target hosts or network environment from the Security Evaluation Rule DB and the Security Policy DB, and stores them in Customized Rule DB. Customized rules in the DB are referenced for the security evaluation. If the customized rules already exist and if they are made for the same host or network environment from the same security evaluation rules and security policy, the analyzer omits the Rule DB and Policy DB access and uses the customized rules that already exist. This reduces the DB access and search time. If there is any conflict between security evaluation rules and security policy, security policy has the preference.

Detail architecture and processing flow of DSS/NSM are as follows.

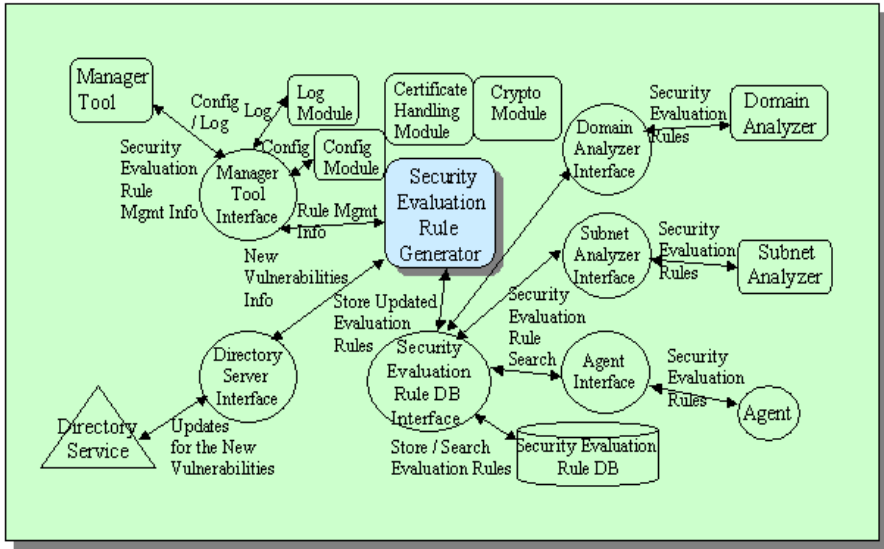


Fig. 2. Architecture of the Security Evaluation Rule Manager

3.3.1 Generation of the Security Evaluation Rules by the Security Evaluation Rule Manager

Fig. 2 illustrates the architecture of the Security Evaluation Rule Manager. Security Evaluation Rule Manager gets the new vulnerability information from the external trusted directory service frequently. Security Evaluation Rule Generator generates the security evaluation rules from the vulnerability information by security manager’s command from the Manager Tool, and stores them in the Security Evaluation Rule DB. A security manager generates, updates and deletes the security evaluation rules.

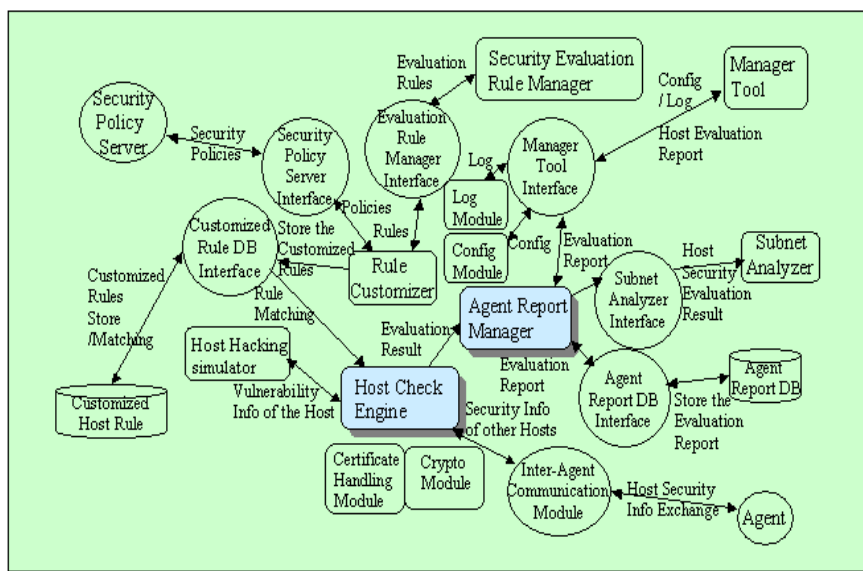
Security Evaluation Rule Manager helps the security manager to generate and manage the evaluation rules. Security Evaluation of Agents, Subnet Analyzers, and Domain Analyzer is based on the security evaluation rules. If an analyzer requests the security evaluation rule, Rule Manager searches for the rule from the Rule DB and return it using the analyzer interface.

3.3.2 Security Evaluation of a Host Using Agent

Fig. 3 illustrates the architecture of the Agent. Rule Customizer of each Agent selects the rules and policy that are suitable to the target host from Security Evaluation Rule Manager and Security Policy Server, and stores them in the Customized Rule DB. If the Customized Host Rules already exist and if they are made for the same host environment from the same security evaluation rules and Security Policy, the Agent omits the Rule DB and Policy DB access and uses the customized rules that already exist.

Host Check Engine evaluates the security of the target host based on the Customized Host Rules. It may exchange the host security information with other Agents to find out the vulnerability that can be checked using the information of other hosts.

It checks the vulnerability of the target host using Host Hacking Simulator. In this case, Host Hacking Simulator has to leave a message to the host to distinguish the test from real hacking. Host Check Engine accesses the low-level detail of the target host's operating system, specific services, and configuration details. It analyzes the system configuration error, bug patch status, signs of the hacker's intrusion, system file integrity and the existence of the Trojan horses, viruses or sniffers. It also checks whether the host obeys the security policy or not.

**Fig. 3.** Architecture of the Agent

Agent Report Manager generates the host security evaluation result report that describes the information related to vulnerability based on the security evaluation result from Host Check Engine. It stores the report in the Agent Report DB and sends the report to the Subnet Analyzer that is in the same subnet.

3.3.3 Security Evaluation of a Subnet Using Subnet Analyzer

Fig. 4 illustrates the architecture of the Subnet Analyzer. If the Customized Subnet Rule does not exist in the Customized Rule DB, Rule Customizer of the Subnet Analyzer selects the rules and policy that are suitable to the target subnet from Security Evaluation Rule Manager and Security Policy Server. It stores them in the Customized Subnet Rule DB.

Subnet Check Engine evaluates the security of the target subnet based on the Customized Subnet Rules. It can exchange the subnet security information with other Subnet Analyzers to find out the vulnerability that can be checked using the information of other subnet.

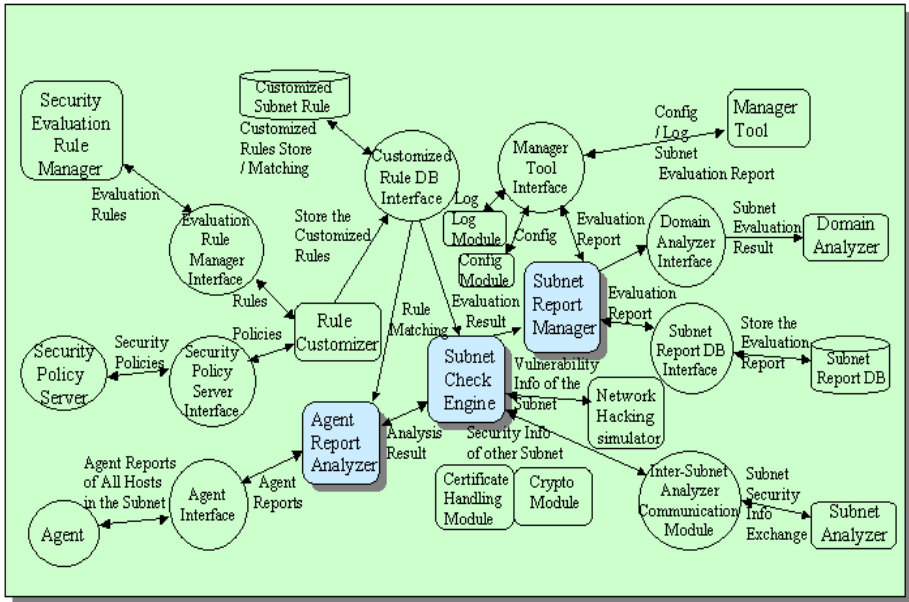


Fig. 4. Architecture of the Subnet Analyzer

It checks the vulnerability of the remote hosts using Network Hacking Simulator. In this case, Network Hacking Simulator has to leave a message to the remote host to distinguish the test from real hacking.

Subnet Check Engine evaluates the security of a subnet from the viewpoint of a network user. Each Subnet Analyzer analyzes vulnerability of the network services and daemons. It analyzes the host security evaluation reports from each agent in the subnet and finds out the vulnerability that can be only checked with the information of several hosts that are part of the distributed environments using Agent Report Analyzer. It also checks whether the subnet obeys the security policy or not.

Subnet Report Manager generates the subnet security evaluation result report that describes the information related to the vulnerability based on the security evaluation result from Subnet Check Engine. It stores the report in the Subnet Report DB and sends the report to the Domain Analyzer.

3.3.4 Decision-Making about Security Management of the Network Domain Using Domain Analyzer

Fig. 5 illustrates the architecture of the Domain Analyzer. Rule Customizer of the Domain Analyzer selects the rules and policy that are necessary for Domain Analyzer from the Security Evaluation Rule Manager and the Security Policy Server, and stores them in the Customized Domain Rule DB.

Security Management Decision Engine evaluates the security of the entire network domain based on the Customized Domain Rules and security evaluation results from each Agents and Subnet Analyzers. It analyzes the subnet security evaluation reports

from each subnet in the domain and finds out the vulnerability that can be checked with the information of several subnet using Subnet Report Analyzer.

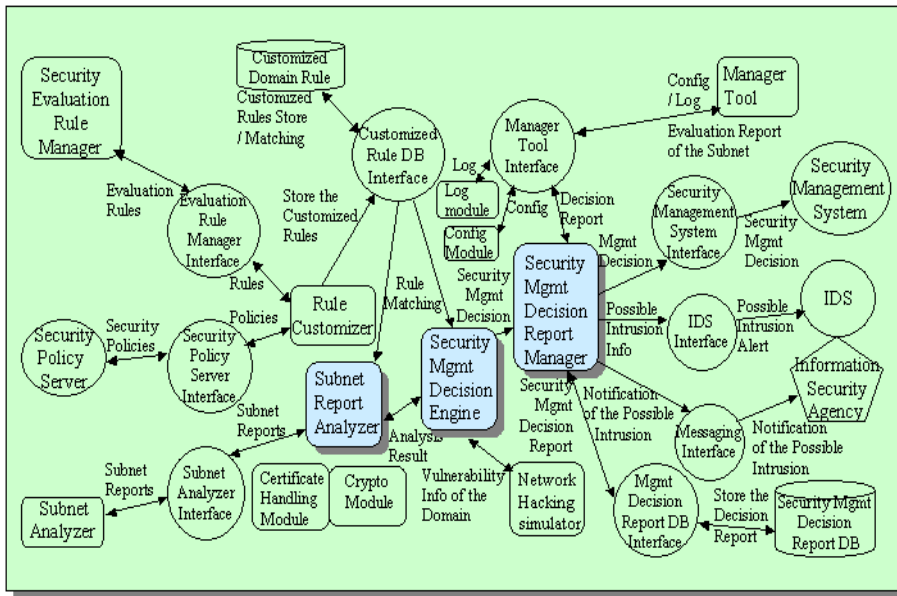


Fig. 5. Architecture of the Domain Analyzer

It also finds out the correlation between the vulnerability. It combines the duplicated results and removes the wrong result from the evaluation results. It predicts the possible intrusion due to the vulnerability analyzed so far. Then it makes decision about security management to prevent the possible intrusion.

Security Management Decision Report Manager generates the Security Management Decision Report based on the security management decision from Security Management Decision Engine. It stores the report in the Security Management Decision Report DB. A security manager can view the report using Manager Tool and use it for decision-making about security management of the network.

Domain Analyzer notifies the decision about security management to security management systems or Intrusion Detection Systems and helps them to prevent the possible intrusion. It requests the proper system configuration change to the security management system and it notifies the possible intrusion to the Intrusion Detection System. If the signs of the serious hacking have been found, it sends the notification mail to the Information Security Agency. It also sends the notification mail to the system manager of the host that has the security weaknesses.

3.3.5 Report View and System Management Using Manager Tool

Fig. 6 illustrates the architecture of the Manager Tool. A security manager can configure each component of DSS/NSM remotely. He can centrally manage the Security

Evaluation Rules and Security Policy. Manager Tool helps the security manager to view the evaluation reports in various formats.

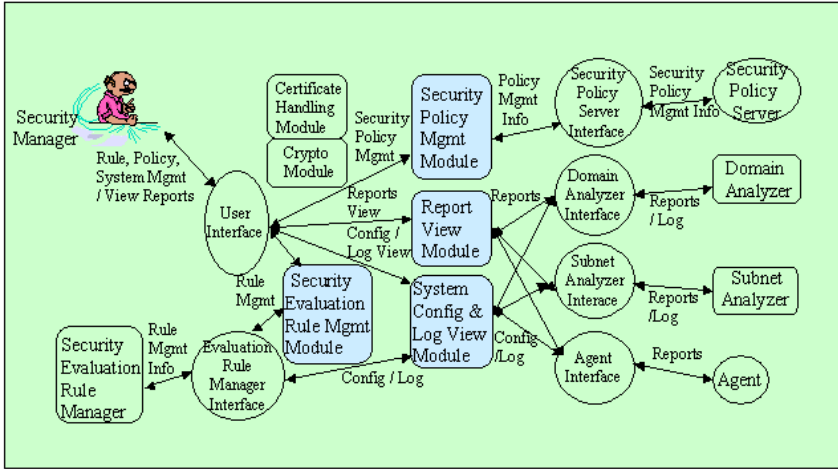


Fig. 6. Architecture of the Manager Tool

User Interface calls the proper module to process the manager's requests and displays the result to the manager. A security manager can add, delete, or update the security policy of the Security Policy Server using Security Policy Management Module. He can view reports in various formats using Report View Module. He also can add, delete, or update the security evaluation rules of the Security Evaluation Rule Manager. He can configure the system or view logs using System Configuration & Log View module.

4 Implementation

In this section, we explain our prototype implementation of DSS/NSM. We are at the beginning stage of implementing DSS/NSM and we implemented basic functions of Agents, Analyzers, Manager Tool and Security Evaluation Rule Manager in the first place. Manager Tool has been implemented in Java and other modules have been implemented in C/C++. We tested the prototype on ISCAP, a security platform that is being developed by ETRI.

ISCAP (Internet Secure Connectivity Assurance Platform) is a platform that provides security services at IP layer, and IPsec [13] Protocol Engine that is part of the platform provides confidentiality, integrity, data authentication, access control and anti-replay services to the application layer of Internet services from IP layer. ISCAP consists of three subsystems, IP Secure connectivity Host System (ISHS), IP Secure connectivity Gateway System (ISGS) and IP Security Control System (ISCS), and each subsystem is connected to Internet and communicates with each other using IP packets.

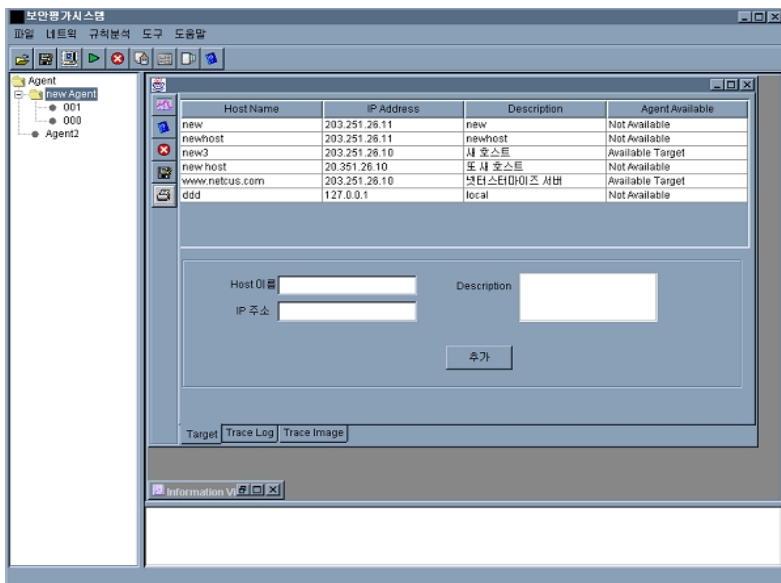


Fig. 7. Selection of Target Hosts to Perform Security Evaluation

Security Evaluation of the systems that provide such security services is much more important than that of other systems, because the failure of the systems makes the entire network domain insecure. Security managers have to check whether the hosts that are part of the security service platform have the vulnerability or not frequently. He also has to check whether the security services provided by the platform work properly. Therefore, we chose the network protected by ISCAP as the first test target.

The prototype of DSS/NSM analyzes each host of ISCAP and finds out typical UNIX vulnerabilities from it. It also checks whether ISCAP provides proper IPsec services to the network or not. To do this, it sniffs the packets generated by the protocols that are used by ISCAP, such as AH [14], ESP [15], ISAKMP [16], SPP [17], SNMP, IP, ICMP, and forges them, and sees what happens to the systems. We have added several modules to the prototype for this test that are specific to IPsec.

Fig. 7 shows the user interface for selecting target hosts to perform Security Evaluation. A security manager can select target hosts using Manager Tool. He also can add, delete or modify Security Evaluation Rules with it.

Once the Security Evaluation process starts, each Agent and Subnet Analyzer start to evaluate the security of each host and subnet protected by ISCAP using Security Evaluation Rules. Then they send the evaluation results to Domain Analyzer. Fig. 8 shows the user interface provided by Manager Tool that displays the results from the Security Evaluation of IPsec services. Domain Analyzer gathers all the results from other analyzers and agents, integrates them and stores the security vulnerabilities of the network in the Report DB. A security manager can retrieve the Report DB using Manager Tool.

Our prototype can be used to analyze the security of other security platforms based on IPsec and it can be extended to a general-purpose security analysis tool.

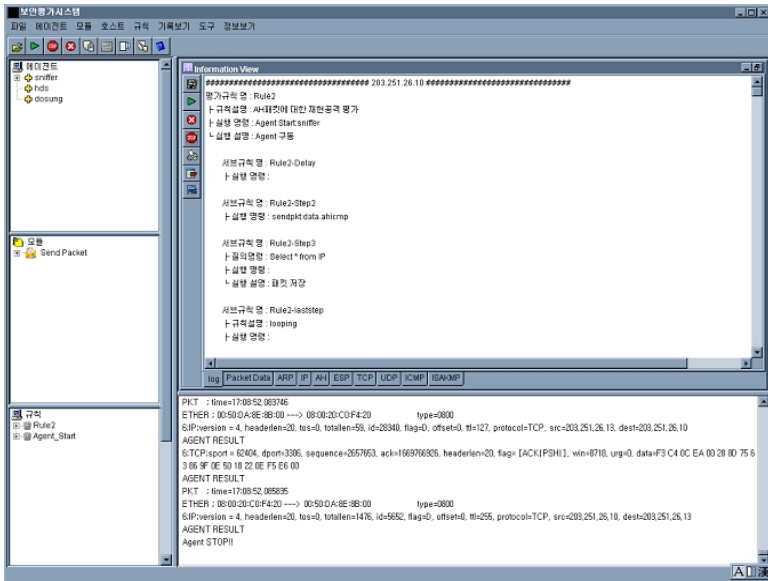


Fig. 8. Security Evaluation using Security Evaluation Rules

5 Conclusions

In this paper we have analyzed requirements of DSS/NSM that automates the security evaluation of a network consists of various components and that supports decision-making about security management to prevent the intrusion, and we have proposed a design for it which satisfies the requirements. We also provided a prototype that implements basic functions of DSS/NSM.

In our design, DSS/NSM gets new vulnerability information from the directory service frequently and updates the Security Evaluation Rule based on it. Therefore DSS/NSM can cope with the new vulnerability rapidly. The system analyzes the security of each host and subnet, and it evaluates the security of the entire network domain based on the analysis result of each component. DSS/NSM then supports decision-making about security management based on the evaluation result to prevent the intrusion in advance.

In the design, security analysis processes are distributed over several analyzers. Therefore the processing load of each analyzer is relatively small. Overall performance of DSS/NSM can be preserved from decrease by adding new Agents and Subnet Analyzers to the network domain even though the number of hosts in the network domain increases. That is, DSS/NSM is scalable. DSS/NSM requires relatively small network traffic because not all the host or network information is sent to the Domain Analyzer. Agents or Subnet Analyzers send the security evaluation report to the Domain Analyzer and it describes only information related to the security evaluation of the host or the subnet. It also greatly reduces the amount of data that the Domain Analyzer has to analyze. Therefore, DSS/NSM is suitable for large network.

DSS/NSM evaluates the security of the network based on the security policy of the Security Policy Server. Therefore, it is easy to check whether the various components of the network obey the security policy of the network domain or not.

DSS/NSM references security policy for the security evaluation but so far the research on the security policy system is not enough and further research is required. Artificial Intelligence technology such as Expert System has to be considered for the accurate and intelligent decision support for the security management. AI technology is also needed for the prediction of the intrusion caused by specific vulnerability. Our prototype implementation just provides several basic functions of our design and it has to be upgraded to reflect and verify all idea of our design.

References

1. ISS, "Network and Host-based Vulnerability Assessment, " <http://documents.iss.net/whitepapers/nva.pdf>
2. ISS, "Securing Operating Platforms: A solution for tightening system security," 1997.1.
3. AXENT Home Page, <http://www.axent.com>
4. ISS Home Page, <http://www.iss.net>
5. Kane Security Analyst Product Home Page, <http://www.mantech.co.kr/ksa.html>
6. J.S. Lee, S.C. Kim, J.T. Lee, K.B. Kim and S.W. Sohn, "Design of the Security Evaluation System for the prevention of hacking incidents under large-scale network environment," *Proceedings of the 12th Workshop on Information Security and Cryptography*, pp. 160-176, Chun-An, 2000.9.
7. J.S. Lee, S.C. Kim, K.B. Kim and S.W. Sohn, "Design of the Security Evaluation System for the automatic security analysis of the large-scale network," *Proceedings of the 5th Conference on Communication Software*, pp. 172-176, Sok-Cho, 2000.7.
8. Larry J. Hughes, Jr., *Actually Useful Internet Security Techniques*, New Riders Publishing, 1995.
9. S. J. Shin, J. W. Yoon and B. M. Lee, "A Prototype Design of Expert System for Automated Risk Analysis tool," *Proceedings of the 10th Workshop on Information Security and Cryptography*, pp. 383-395, 1998.
10. S.W. Kim, H. J. Jang and B. Park, "Dynamic Monitoring based on Security Agent," *Proceedings of the 10th Workshop on Information Security and Cryptography*, pp.518-530, 1998.
11. Sundaram. Aurobindo, "An Introduction to Intrusion Detection," ACM CROSSROADS Issue 2.4, 1996.4.
12. Simson Garfinkel & Gene Spafford, *Practical UNIX & Internet Security*, O'REILLY, Second Edition, April 1996.
13. IETF RFC2401, "Security Architecture for the Internet Protocol", November 1998.
14. IETF RFC2402, "IP Authentication Header (AH)", November 1998.
15. IETF RFC2406, "IP Encapsulating Security Payload (ESP)", November 1998.
16. IETF RFC2408, "ISAKMP", November 1998.
17. IETF Internet-Draft, "Security Policy Protocol".

Measuring False-Positive by Automated Real-Time Correlated Hacking Behavior Analysis

Jia Wang and Insup Lee

Department of Computer and Information Science,
University of Pennsylvania
Philadelphia, Pennsylvania, USA
jia@saul.cis.upenn.edu, lee@cis.upenn.edu

Abstract. To solve the contradiction between the trend of more distributed network architecture and the demanding for more centralized correlated analysis to detect more complicated attacks from Intrusion Detection System (IDS), we first proposed in this paper an IDS architecture framework, which could collect relevant detected alert data from distributed diverse IDSes into one or more centralized point(s), and then efficient correlation analysis would be processed on shared data, after that, the meaningful and supportive knowledge rules from analysis results were be generated and automatically pushed back to each subscribed local IDS on scheduled time or even in real time, so that local IDS could utilize these rules to analyze new coming traffic. We also defined the XML format for those knowledge rule information generated by our hacking behavior correlation algorithms. We then presented seven mathematical algorithms on correlated hacking behavior analysis. In order for local IDS to effectively measure the false positive possibility of a new coming alert, we introduced three different approaches using some data mining and statistic models, including 1-Rule, Bagging Method and Native Bayer Method. By applying these methods to utilize and analyze the collected correlated knowledge rules, we could derive quite good quality of true attack confidence value for each coming detected alert. We also developed a simulation program implementing all these correlation algorithms and all those data mining and statistic models. We simply tested these algorithms with MIT Lincoln Lab's 1999 IDS evaluation data, and concluded that by utilizing these preliminary results, local IDS subscribed to this framework could derive a certain measurement of how confident an alarm is true attack in real time manner and even lower false positive rate if certain threshold applied.

1 Introduction

1.1 What Is False Positive

As described by Cisco Systems, False positives (benign triggers) occur when the Intrusion Detection System (IDS) report certain benign activity as malicious, requiring human intervention to diagnose the event. Obviously, a large number of false positives can significantly drain resources, and the specialized skills required to analyze them are costly and difficult to find. Unfortunately, due to the nature of the

signatures that IDSes use to detect malicious activity, it is virtually impossible to completely eliminate false positives without severely degrading the effectiveness of the IDS or severely disrupting an organization's computing infrastructure (such as hosts and networks). False negatives, on the other hand, occur when the IDS does not detect and report actual malicious activity. False negative can be reduced by simply adding more detection or defense layers, however, layers will not reduce overall false positive rates.

Most intrusion detection systems have to allow system administrators to adjust an alert threshold that effectively trades off false negatives for false positives. By lowering the threshold, an administrator can discover more attacks, but will likely have to see more false alarms. Similarly, an administrator can raise the threshold to reduce false alarms, but this will also likely cause the system to miss some additional attacks. Knowledge has to be learned to tune the thresholds to match the amount of available effort and desired level of security. Since human beings are operating IDS, that system with above hundreds of false positive per day will make it excessively expensive to deploy, even with high detection accuracy.

1.2 Challenges

Most of the IDSes installed are independently operated in terms of traffic or behavior analysis, however, hacking information detected from relevant hosts or paths are somewhat related to each other. And widely distributed heterogeneous computer systems with all kinds of different vulnerabilities caused IDSes very difficult to be effective, especially with huge amount of volatile raw data traffic and lack of communication and trust among detection entities. The ability to collect, assimilate, correlate, and analyze information emanating from diverse sources in real time becomes essential. Also, the attacks are becoming more sophisticated and organized, and today it is still a very difficult task to protect our network from many complicated and coordinated attacks. Knowing that hacking information of different time periods and locations are related to each other, so correlation over both time and space dimensions could be analyzed to improve the accuracy and efficiency of the detection. The distributed IDS is trying to address some of these challenges, however, the distribution nature is contradict to the demanding of more centralized study on overall data collections.

1.3 Related Work on IDS

We learnt some distributed IDSes related to our work [4], for example, EMERALD from SRI International [11], provides an architecture to facilitate enterprise-wide deployment and configuration of intrusion detectors, which support distribution, correlation and real time reaction; Another system called the DIDS uses simple monitor as its client; Also the system called JAM from Columbia University uses smart agent and machine learning technology [15]. Other systems including Ji-Nao System from North Carolina State University [8], GrIDS system from University of California at Davis [12], AXENT Multi-tiered IDS from AXENT Technologies [10], and Distributed Immunology IDS from University of New Mexico [1]. All these

systems are distributed, using different architecture models, client features, and handling different correlation analysis. Also a new framework, MADAM ID, for Mining Audit Data for Automated Models for Intrusion Detection from NCSU and Columbia University [13] uses data mining algorithms to compute activity patterns from system audit data and extracts predictive features from the patterns. It focuses on developing methods of algorithms and techniques for constructing intrusion detection models. Our proposed new distributed framework presented in this paper is featured by its full function IDS as local client, central correlation analysis focus on hacking behaviors, instead of module driven by supporting IDS with new detecting algorithms and patterns, we are kind of data driven by generating high efficient data central correlated knowledge rules directly, by applying two way communication protocols between local client and central point, the IDS could share those knowledge rules and achieve real time reaction and extensibility.

2 New Solution

2.1 Overall Architecture

This architecture provides a framework to integrate any kinds of IDS into an open system. With efficient and secure communication protocol defined to exchange data between subscribed IDS and centralized analyzing platform, hacking event and information are collected in standard format and analyzed using certain correlation analysis models. The knowledge rules of analysis results will then be pushed back to each IDS and thus support the local real time detection. Different kind of IDSes could be freely plugged into this framework, as indicated in the Fig. 1.

DSAP is a cooperative framework that allows interactions to take place among all the subscribed ID units. The scope of DSAP could be flexible, like LAN or WAN. Heterogeneous subscribed IDSes could be within the same network monitoring same traffic or in different geographic locations monitoring different but related traffic. By effectively integrating detected alerts from different types of ID systems, It could achieve goals towards measuring and reducing false alarms or other useful purposes. To be more reliable, the DSAP could also maintain one or multiple central point(s) to collect and analyze data.

ASR is defined as extensible knowledge rule sets that are meaningful to local IDS. They are derived by DSAP after processing all the hacking information collected from subscribed IDSes, and are pushed back to each local IDS in scheduled or real time manner. Any kind of algorithms could be implemented to support ASR, including data mining, signal processing, control theory, and artificial intelligence, etc. We are going to implement seven correlation algorithms based on hacking behavior analysis in this paper, and each of them will generate useful knowledge rules. Each subscribed IDS could specify the criteria about what kind of ASR knowledge it interested, so only those related rules will be pushed back from DSAP. To the most case, all the ASR rules in DSAP could be pushed to all the subscribed IDS as required.

HDIS is a common structure holding hacking information detected by different implementations of IDSes sending to DSAP in real time. It has to be compatible, complete and efficient. The research in this area is covered by Intrusion Detection Exchange Format Data Model, which could contribute directly into our framework. In our simulation, to make it simple, we just used the data format defined by DARPA Off-line Intrusion Detection Evaluation in MIT[5], which including alert's (date, method, category, time, duration, source, destination).

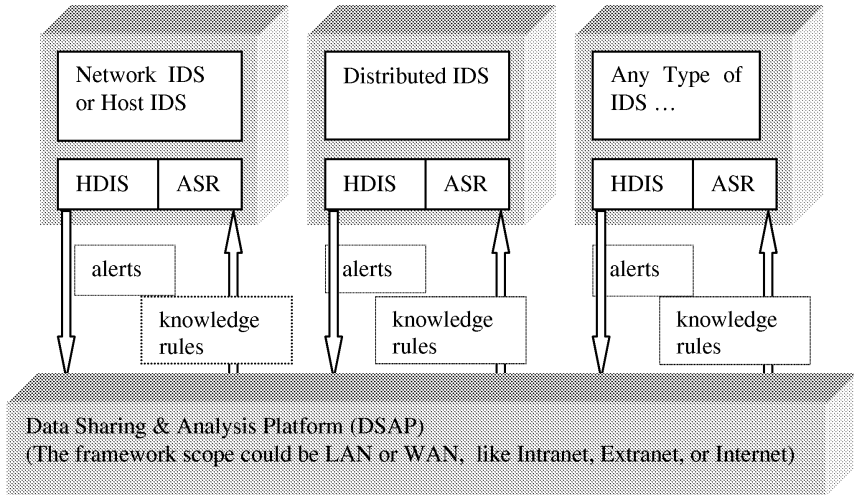


Fig. 1.

Notation: HDIS - Hackers Detection Information Structure
 ASR - Analysis Support Resource
 DSAP - Data Sharing and Analysis Platform

The two-way asynchronous communication mechanism defined in the DSAP architecture could represent both HDIS and ASR data using XML format, which is becoming a standard markup language for communication data containing structured information among different applications across the network. The Intrusion Detection Exchange Format Work Group (IDWG) from Internet Engineering Task Force (IETF) has done some very useful research in this area [2], [3]. Both the communication protocol and HDIS format could be covered by their research. We define ASR Data Format in XML Document Type Definition (DTD) as:

```
<?xml version="1.0" ?>

<!DOCTYPE asr_knowledge [

<!ELEMENT asr_knowledge (asr_header, asr_rules)>

<!ELEMENT asr_header (ids_id, date_time)>
```

```

<!ELEMENT ids_id (#PCDATA)>

<!ELEMENT date_time (#PCDATA)>

<!ELEMENT asr_rules (rule*)>

<!ELEMENT rule (type, argument, value)>

<!ELEMENT type (#PCDATA)>

<!ELEMENT argument EMPTY>

<!--ATTLIST argument

src CDATA #IMPLIED

dest CDATA #IMPLIED

method CDATA #IMPLIED

vulnerability CDATA #IMPLIED-->

<!ELEMENT value (#CDATA)> ]>

```

2.2 What's New

What we presented in this paper is a new high level IDS framework; we try to address many of the challenges currently facing the IDS technology, especially the distributed ones. Following advantages could be achieved altogether in our new framework. Also, we implemented seven correlated hacking behavior analysis algorithms (see 3.1). Given the ability to identify an attacker from his prior methods of operation, and given an understanding of the attacker's prior goals, it may be possible to infer where his current activity is leading and analyzing his motivation and predicting intruder's intent. By applying them into our new framework, we could measure the true attack confidence value of each new detected alert, and even lower subscribed IDS's false positive rate if necessary (see 3.2).

2.2.1 Heterogeneous Open Subscription Architecture

Different IDSeS with heterogeneous matching mechanisms or architecture models can subscribe into this framework simply by complying with HDIS and implementing the communication protocol. It is flexible, open and even secure if the authentication and authorization procedures are applied in subscription process. Once an IDS subscribes into framework, it will start to utilize knowledge rules from the framework. It is obvious that heterogeneous IDS could be benefited from data sharing of each other. Actually sharing information from heterogeneous IDSeS could be critical in reducing high false alarm rates.

2.2.2 Real Time Reaction

Once the IDS subscribes into the DSAP framework, it will start to receive ASR rules and build up its local knowledge base. The local knowledge base will keep growing and refreshing as long as the IDS stays within the framework. So that the IDS can utilize the local ASR knowledge base immediately. Real time reaction also owes credits to local ID system's full functionality, Since only detected alerts are taken into account, most noise (normal traffic) is filtered out locally. This become very useful because otherwise the huge volume, volatility and noisiness of data with false positives generated by ID systems will flood the central point in DSAP, especially for large distributed networks. This way, our solution makes local IDS more intelligent and more efficient.

2.2.3 Centralized Extensive Correlation Analysis Model

The centralized correlation analysis could benefit each subscribed IDS in the framework. It could find certain intrusions, which may be difficult to be detected by individual IDS alone. The advantage of this ASR model is that it is open and extensive, we can invent new models to add into it without impact previous knowledge rules. So that it is very easy to integrate any new breakthroughs from many areas like data mining, AI, etc. into IDS technology.

2.2.4 Efficient Knowledge Rules PushBack

Today's IDS technology cannot nicely merge the distributed architecture and centralized correlation together. Centralized one-way data collection and event processing IDS architecture is a passive information processing paradigm. It faces considerable limitations when trying to detect increasingly sophisticated attacks in large and complex networks. With DSAP framework, a two-way secure communication mechanism is introduced to collect detected alerts and push back knowledge rules in both directions. We implemented this data only knowledge rules to be both very efficient and extensive, thus could achieve the goal of combining distribution and centralization without so many performance concerns.

2.3 How It Works

To set up the DSAP includes defining the HIDS format, implementing communication protocols and developing ASR models. After that, individual IDS could subscribe to the DSAP and take some time to train itself to tune the constant value set and the threshold properly. Once ready, subscribed IDS starts to function and send its detected alerts to central point of DSAP using HIDS format. The DSAP is keep collecting detected alert data from each IDS; many correlation algorithms will be processed on regular bases, for example, in our case, seven hacking behavior analysis algorithms will be processed. And then the meaningful and supportive results of ASR knowledge will be generated and automatically pushed back to each subscribed IDS in scheduled or real time. Each IDS will keep the received ASR in its knowledge base, and utilize the information locally for any new coming traffic in real time. The asynchronous communication protocol will not guarantee the local IDS always receiving the most recent ASR in time, however, since we are using the ASR for a long time, the effect of this kind of delay compared with the knowledge's long

history could be minimum. The IDS can now calculate the True Attack Confidence Value using its local ASR knowledge base. This value can be used to measure the real-attack confidence of each alert and to lower the false positive rate if apply a threshold.

2.3.1 An Example in Detail

In one of our simulation tests using data from 1999 MIT DARPA Project, four different type of ID systems are subscribed monitoring the same network, namely IDS-1, IDS-2, IDS-3 and IDS-4. To make it simple, let's just apply one of the hacking behavior algorithms called "Victim Vulnerability Abuse Rate" to the analysis (see 3.1.2), and assume that this ASR is the only contributor for IDS-1 to calculate the True Attack Confidence Value, which means that we set the weight constant VV to be 1, and others to be 0 (see 3.2.1); And we also assume that the threshold based on True Attack Confidence Value already tuned to be 40% on IDS-1 during the training period based on previous data.

The IDS-1 reported following two alerts in format (date, method, category, time, duration, source, destination) as:

```
alert1: 03/29/99, Passwordguessing, r2l, 16:11:15,
00:00:01, 206.186.80.111, 172.016.112.100
```

```
alert2: 03/30/99, ProcessTable, DOS, 17:49:15,
00:03:00, 208.240.124.083, 172.016.114.050
```

From IDS-1 only, we cannot tell whether these two alerts are false positives or not. However, since we have other three IDSes also report alerts to DSAP, the algorithm actually found that on victim 172.016.114.050, there were already three DOS alerts out of total five alerts reported to DSAP before alert2 by other IDSes, these three alerts happened at 8:14:54, 15:51:16 and 15:51:18 on 03/30/99, so that the generated ASR for this algorithm at time of 15:51:18 is $\{ASRVV_R(172.016.114.050, DOS)=60\%$ } (see 3.2).

When IDS-1 detected alert1, as we know that the same ASR, $\{ASRVV_R(172.016.112.100, r2l)=0\%$ }, which was below the threshold, the IDS-1 reported alert1 to be false positive. Suppose the DSAP could push the $ASRVV_R$ back to IDS-1 between 15:51:18 and 17:49:15 (nearly two hours gap), so that when alert2 was detected, its local knowledge base was updated, and IDS-1 calculated the confidence value to be 60%, which was higher than threshold, so it reported alert2 to be true alert. These results matched the actual situation based on simulation design. The false positive rate on IDS-1 reduced from 50% to 0 after using our solution.

3 Hacking Behavior Analysis

3.1 Correlated Hacking Behavior Algorithms

We are going to explore some correlated algorithms about computer hacking behavior by sharing alert's coordinated relationships, such as common points of origin, common protocols, common targets, correlated attack timing and rate, etc. The formal mathematical expressions are also presented.

3.1.0 Basic Definitions

We first need to define some basic terms to describe the whole framework, each IDS and each alert detected by IDS.

Definition 0.1: We define the whole-distributed system as a finite set D . And each participated IDS is an element $d_i \in D$. The size of D is n , that is $|D| = n$, and $D = \bigcup_{i=1}^n d_i$

Definition 0.2: We define an alert a_i to be a detected alarm from d_i , and each a_i is represented as a 6-tuple

$$a_i(t) = (i, t, s_i, m_i, e_i, v_i)$$

where i is the i th IDS in D ; t is the alarm happening time; s_i is the source address; m_i is the attacking method; e_i is the destination address; v_i is the vulnerability

Definition 0.3: We define an alert array $S(d_i, t)$, with total of m alarms detected by IDS d_i before time t

$$S(d_i, t) = \{ a_i(t_1), a_i(t_2), a_i(t_3), \dots, a_i(t_m) \},$$

where $t_1 \leq t_2 \leq t_3 \leq \dots \leq t_m \leq t < t_{m+1}$; $|S(d_i, t)| = m$;

Definition 0.4: We define an alert array $S(D, t)$, with all the alarms detected by $\forall d_i \in D$ before time t

$$S(D, t) = \bigcup_{i=1}^n (S(d_i, t)),$$

where $|S(D, t)| = \sum_{i=1}^n (|S(d_i, t)|)$;

3.1.1 Attack Count from Same Location

We realize that hacker's preparation before attacking involves finding target network structure and identifying the most vulnerable hosts within the target network. And most attacks start with many scanning and probe activities. And those pre-attack probe requests would hit most of the hosts on the whole target network. In this situation, Many ID systems all over the target network will detect these scanning behaviors and generate scanning alerts. Although these alerts might occur to different victims, yet they are very possible to come from the same source groups.

Definition 1.1: For each alert $a_i(t)$, We define function for AL_Count as

$$F_{AL_C}(a_i(t)) = \sum_{b \in S(D, t)} \phi(b, a_i(t)) ;$$

$$\text{where } a_i(t) = (i, t, s_i, m_i, e_i, v_i)$$

$$b = (i_b, t_b, s_b, m_b, e_b, v_b)$$

$$\text{Function: } \phi(b, a_i(t)) = \{1 | s_i = s_b, 0 | s_i \neq s_b\}$$

Definition 1.2: For each alert $a_i(t)$, We define function for AL_Count Rate as

$$F_{AL_R}(a_i(t)) = F_{AL_C}(a_i(t)) / |S(D, t)| ;$$

Definition 1.3: We define ASR derived from F_{AL_R} as

$$ASR_{AL_R}(s_i) = F_{AL_R}(a_i(t))$$

3.1.2 Victim Vulnerability Abuse Rate

After a system's certain vulnerability was exploited, there is a big possibility that the similar attacks making use of this same vulnerability would happen again. This is because the same hacker could come back, or other hackers could find the same vulnerability and to make use of, or maybe because hackers share their success to show off their capabilities, etc. And in some cases, this vulnerability abuse analysis can also apply to different hosts, yet with same hardware system and operating system from the same victim's network. Thus if IDS detects an attack making use of the same vulnerability previous already happened to the victim, it is more suspicious to us.

Definition 2.1: For each alert $a_i(t)$, We define function for VV_Rate as

$$F_{VV_R}(a_i(t)) = F_{VV_C}(a_i(t)) / F_{VL_C}(a_i(t)) ;$$

$$F_{VL_C}(a_i(t)) = \sum_{b \in S(D, t)} \gamma(b, a_i(t)) ;$$

where $a_i(t) = (i, t, s_i, m_i, e_i, v_i)$

$b = (i_b, t_b, s_b, m_b, e_b, v_b)$

Function: $\gamma(b, a_i(t)) = \{1 | e_i=e_b; 0 | e_i \neq e_b\}$

$F_{VV_C}(a_i(t)) = \sum_{v_b \in S(D,t)} \eta(b, a_i(t))$;

where $a_i(t) = (i, t, s_i, m_i, e_i, v_i)$

$b = (i_b, t_b, s_b, m_b, e_b, v_b)$

Function: $\eta(b, a_i(t)) = \{1 | e_i=e_b \ \&\& \ v_i=v_b; 0 | e_i \neq e_b \ || \ v_i \neq v_b\}$

Definition 2.2: We define ASR derived from F_{VV_R} as

$ASR_{VV_R}(e_i, v_i) = F_{VV_R}(a_i(t))$

3.1.3 Attacking Method Frequent Use Rate

Many hackers utilize complicated toolset during hacking, especially when exploiting a known and relatively standard security holes of the target. And almost all hackers spend a lot of time and energy to play with existing hacking tools or invent new ones. Thus it is reasonable that certain hackers are willing to use certain hacking methods. The correlation information between source address and attack method can be also very helpful to detect this behavior, and help to identify a real attack at an earlier stage.

Definition 3.1: For each alert $a_i(t)$, We define function for AM_Rate as

$F_{AM_R}(a_i(t)) = F_{AM_C}(a_i(t)) / F_{AL_C}(a_i(t))$;

$F_{AL_C}(a_i(t))$ (definition 1.1)

$F_{AM_C}(a_i(t)) = \sum_{v_b \in S(D,t)} \iota(b, a_i(t))$;

where $a_i(t) = (i, t, s_i, m_i, e_i, v_i)$

$b = (i_b, t_b, s_b, m_b, e_b, v_b)$

Function: $\iota(b, a_i(t)) = \{1 | s_i=s_b \ \&\& \ m_i=m_b; 0 | s_i \neq s_b \ || \ m_i \neq m_b \}$

Definition 3.2: We define ASR derived from F_{AM_R} as

$ASR_{AM_R}(s_i, m_i) = F_{AM_R}(a_i(t))$

3.1.4 Concurrent Hacking Frequency

There are two correlation models over time concurrence that are interesting to us and could be helpful to support local detection. Many hackers utilize advanced fast tools during hacking. And if there are a lot of hacking activities going on during a relatively short period of time, it is possible that these attacks are related to each other, maybe initiated by the same hacker. The situation could either be attacking multiple victims from same source address, or attacking same victim address from multiple sources.

Definition 4.1: For each alert $a_i(t)$, We define function for AC_Frequency as

$$F_{AC_F}(a_i(t), T_{HTP}) = \sum_{c \in T(D,t)} \phi(c, a_i(t)) ;$$

where T_{HTP} is Hacking Time Period Constant

$$T(D,t) = S(D,t) - S(D, (t-T_{HTP}))$$

$$a_i(t) = (i, t, s_i, m_i, e_i, v_i)$$

$$c = (i_c, t_c, s_c, m_c, e_c, v_c)$$

$$\text{Function: } \phi(c, a_i(t)) = \{1 | s_i = s_c, 0 | s_i \neq s_c\}$$

Definition 4.2: For each alert $a_i(t)$, We define function for AC_Frequency Rate as

$$F_{AC_R}(a_i(t), T_{HTP}) = F_{AC_F}(a_i(t)) / |T(D, t)| ;$$

Definition 4.3: For each alert $a_i(t)$, We define function for VC_Frequency as

$$F_{VC_F}(a_i(t), T_{HTP}) = \sum_{c \in T(D,t)} \lambda(c, a_i(t)) ;$$

where T_{HTP} is Hacking Time Period Constant

$$T(D,t) = S(D,t) - S(D, (t-T_{HTP}))$$

$$a_i(t) = (i, t, s_i, m_i, e_i, v_i)$$

$$c = (i_c, t_c, s_c, m_c, e_c, v_c)$$

$$\text{Function: } \lambda(c, a_i(t)) = \{1 | e_i = e_c, 0 | e_i \neq e_c\}$$

Definition 4.4: For each alert $a_i(t)$, We define function for VC_Frequency Rate as

$$F_{VC_R} (a_i(t), T_{HTP}) = F_{VC_F} (a_i(t)) / |T(D, t)| ;$$

Definition 4.5: We define ASR derived from F_{AC_R} & F_{VC_R} as

$$ASR_{AC_R} (s_i) = F_{AC_R} (a_i(t), T_{HTP})$$

$$ASR_{VC_R} (e_i) = F_{VC_R} (a_i(t), T_{HTP})$$

3.1.5 Hacking Behaviors over Timing Regularity

We know that many hackers spend many of their spare times on hacking and they could even keep a relatively regular hacking schedule. To extract this information, we could defined following Timing Period as an example {Morning: 6:00am - 12:00noon, Afternoon: 12:00noon - 6:00pm, Evening: 6:00pm - 12:00 midnight, LateNight: 12:00midnight - 6:00am}. And if detected alert from certain source or victim address have consistent occurrences previously at the same time of the day, it could be noticed.

Definition 5.1: For each alert $a_i(t)$, We define function for AR_Rate as

$$F_{AR_R} (a_i(t), \Gamma) = F_{AR_C} (a_i(t), \Gamma) / F_{AL_C} (a_i(t));$$

$$F_{AL_C} (a_i(t)) \text{ (definition 1.1)}$$

$$F_{AR_C} (a_i(t), \Gamma) = \sum_{p \in S(D, t)} \kappa(p, a_i(t)) ;$$

where Γ is a Hacking Timing Regularity Period Mapping Function as

$$\Gamma(t) = r_j, \{00:00:00 - 23:59:59\} \rightarrow_r \{ r_1, r_2, \dots r_1 \}$$

example: we define four ($l = 4$) regularity period here:

$$\Gamma(t) = 1, \text{ iff } t \in [\text{midnight}, 6:00:00) \quad \{\text{late night}\}$$

$$\Gamma(t) = 2, \text{ iff } t \in [06:00:00, \text{noon}) \quad \{\text{morning}\}$$

$$\Gamma(t) = 3, \text{ iff } t \in [\text{noon}, 18:00:00) \quad \{\text{afternoon}\}$$

$$\Gamma(t) = 4, \text{ iff } t \in [18:00:00, \text{midnight}) \quad \{\text{evening}\}$$

where $a_i(t) = (i, t, s_i, m_i, e_i, v_i)$

$$p = (i_p, t_p, s_p, m_p, e_p, v_p)$$

Function: $\kappa(p, a_i(t)) = \{1 | s_i = s_c \text{ and } \Gamma(t) = \Gamma(t_p);$
 $0 | \text{otherwise}\}$

Definition 5.2: For each alert $a_i(t)$, We define function for VR_Rate as

$$F_{VR_R}(a_i(t), \Gamma) = F_{VR_C}(a_i(t), \Gamma) / F_{VL_C}(a_i(t));$$

$$F_{VL_C}(a_i(t)) \text{ (definition 2.1)}$$

$$F_{VR_C}(a_i(t), \Gamma) = \sum_{p \in S(D, t)} \mu(p, a_i(t)) ;$$

where Γ (definition 5.1)

$$a_i(t) = (i, t, s_i, m_i, e_i, v_i)$$

$$p = (i_p, t_p, s_p, m_p, e_p, v_p)$$

Function: $\mu(p, a_i(t)) = \{1 | e_i = e_p \text{ and } \Gamma(t) = \Gamma(t_p);$
 $0 | \text{otherwise}\}$

Definition 5.3: We define ASR derived from F_{AR_R} & F_{VR_R} as

$$ASR_{AR_R}(s_i) = F_{AR_R}(a_i(t), \Gamma)$$

$$ASR_{VR_R}(e_i) = F_{VR_R}(a_i(t), \Gamma)$$

3.1.6 Possible Order Correlation Analysis of Hacking Actions

Typically there are multiple ways to invade a system. Nevertheless, it usually requires several actions and tools be applied in a particular logical order to launch a sequence of effective attacks to achieve a particular goal. No matter how the system is compromised, hackers usually take more than one step and more than one hacking action to finish the whole attack. During a real attack, some actions usually precede other ones. And the correlation information of this hacking action order can be helpful to generate alert in the earlier stage. For example the phases could be {(a) reconnaissance, (b) vulnerability identification, (c) penetration, (d) control, (e) embedding, (f) data extraction (g) attack relay}. Further we could even find the order of hacking sub-actions and build up the order correlation in next level detail.

Definition 6.1: For each alert $a_i(t)$, We define function for VP_Rate as

$$F_{VP_R}(a_i(t), \Phi) = F_{VP_C}(a_i(t), \Phi) / F_{VL_C}(a_i(t));$$

$F_{VL_C}(a_i(t))$ (definition 2.1)

$$F_{VP_C}(a_i(t), \Phi) = \sum_{p \in S(D,t)} \kappa(p, a_i(t)) ;$$

where Φ is a Hacking Method Category Mapping Function

$$\Phi(m_i) = c_j, \quad \forall a_i \in S(D, t), \quad a_i(t) = (i, t, s_i, m_i, e_i, v_i)$$

$$\{m_1, m_2, \dots, m_k\} \rightarrow_{\Phi} \{c_1, c_2, \dots, c_l\}, \quad c_1 < c_2 < \dots < c_l$$

example : we define seven (l=7) method categories here:

$$\Phi(m) = 1, \quad \text{iff } m \in \{ \text{reconnaissance} \}$$

$$\Phi(m) = 2, \quad \text{iff } m \in \{ \text{vulnerability identification} \}$$

$$\Phi(m) = 3, \quad \text{iff } m \in \{ \text{penetration} \}$$

$$\Phi(m) = 4, \quad \text{iff } m \in \{ \text{control} \}$$

$$\Phi(m) = 5, \quad \text{iff } m \in \{ \text{embedding} \}$$

$$\Phi(m) = 6, \quad \text{iff } m \in \{ \text{data extraction} \}$$

$$\Phi(m) = 7, \quad \text{iff } m \in \{ \text{attack relay} \}$$

where $a_i(t) = (i, t, s_i, m_i, e_i, v_i)$

$$p = (i_p, t_p, s_p, m_p, e_p, v_p)$$

Function: $\kappa(p, a_i(t)) = \{1 | e_i = e_p \text{ and } \Phi(t_p) \leq \Phi(t),$
 $0 | \text{otherwise}\}$

Definition 6.2: We define ASR derived from F_{VP_R} as

$$ASR_{VP_R}(e_i, m_i) = F_{VP_R}(a_i(t), \Phi)$$

3.1.7 Concurrency of Organized Attacking Locations

Many hackers attack a victim location using multiple already confiscated sources. In order to identify those relationships among those attacking sources, we assume that hackers might use them at the same period of time to attack a victim host. So that we try to find whether any two attacking source locations appeared together within certain time period, if they do show up together as partners very often, which means

they are used together by the same hacker. So that the concurrency between any two different attacking source locations over certain time period could be helpful to support local detection.

Definition 7.0: We define a two-dimensional integer array for concurrency as $C[n][n](t, T_{HCP})$;

where T_{HCP} is Hacking Concurrency Period Constant

$$n = |S(D, t)|, C[i][j]_{init} = 0 \mid i, j = 1 \dots n;$$

$$\forall a_i(t_i) \in S(D, t), \forall a_j(t_j) \in S(D, t), (i \neq j)$$

$$a_i(t_i) = (i, t_i, s_i, m_i, e_i, v_i)$$

$$a_j(t_j) = (j, t_j, s_j, m_j, e_j, v_j)$$

define function $\sigma(s)$ mapping source address into an index:

$$k = \sigma(s), k \in [1, n]$$

$$C[i_s][j_s] += 1, \text{ iff } e_i = e_j \text{ and } |t_i - t_j| \leq T_{HCP}$$

$$i_s = \sigma(s_i), j_s = \sigma(s_j),$$

Definition 7.1: For each alert $a_i(t)$, We define function for VCC_Confidency as

$$F_{VCC_C}(a_i(t), T_{HCP}) = \sum_{v_c \in S(D, t)} C[i][i_c];$$

where T_{HCP} is Hacking Concurrency Period Constant

$$a_i(t) = (i, t, s_i, m_i, e_i, v_i)$$

$$c = (i_c, t_c, s_c, m_c, e_c, v_c)$$

$$i = \sigma(s_i), i_c = \sigma(s_c);$$

Definition 7.2: For each alert $a_i(t)$, We define function for VCC_Confidency Rate as

$$F_{VCC_R}(a_i(t), T_{HCP}) = F_{VCC_C}(a_i(t), T_{HCP}) / \sum_{i, j \in [1, n]} C[i][j];$$

Definition 7.3: We define ASR derived from F_{VCC_R} as

$$ASR_{VCC_R}(s_i, e_i) = F_{VCC_R}(a_i(t), T_{HCP})$$

3.2 True Attack Confidence Value

Each IDS subscribed to the DSAP keeps receiving ASR knowledge rules from the central point of DSAP. In our case, the nine rules are results from seven correlated hacking behavior analysis done at DSAP. Local IDS keeps a knowledge base to accumulate these rules, and use them to measure the false positive rate. Each IDS could calculate the True Attack Confidence value (Val_{TA}) for a detected alert by using the knowledge base. The higher the Val_{TA} is, the more possible this alert is a true attack (NOT a False Positive).

Definition 8.01: We define all kinds of ASR knowledge rules as a finite set A . And each kind of ASR rule is an element $a_i \in A$. The size of A is m , that is $|A| = m$, and $A = \bigcup_{i=1}^m a_i$

For example, in our case,

$$A = \{AL_R, VV_R, AM_R, AC_R, VC_R, AR_R, VR_R, VP_R, VCC_R\}$$

Definition 8.02: We define all the training data alert instances a finite set T . And each instance is an element $t_i \in T$. The size of T is $|T|$, and $T = \bigcup_{i=1}^n t_i$; We also define the set of all the true attack instances within T as T_t , and the set of all the false alarm instances within T as T_f , obvious $T = T_t + T_f$;

Definition 8.03: For each alert $a_i(t)$, we define

$$a_i(t) = (i, t, s_i, m_i, e_i, v_i) \text{ (definition 0.2)}$$

$$Val_{TA}(a_i(t)) = f_{ASRs}$$

$$\begin{aligned} & \{ ASR_{AL_R}(s_i), ASR_{VV_R}(e_i, v_i), ASR_{AM_R}(s_i, m_i), \\ & ASR_{AC_R}(s_i), ASR_{VC_R}(e_i), ASR_{AR_R}(s_i), \\ & ASR_{VR_R}(e_i), ASR_{VP_R}(e_i, m_i), ASR_{VCC_R}(s_i, e_i) \} \end{aligned}$$

Notation: function $f_{\{ASRs\}}$ is discussed later in this paper

Besides the subscribed local IDS could generate useful Val_{TA} to measure the false alarm possibility, the security administrator could also try to set up a threshold value in case of too many alert detected beyond process power. So that, by training each IDS using function f_{ASRs} . A threshold value could be configured to let the IDS only alert when Val_{TA} is higher than the threshold value. During the training period, each IDS subscribed to DSAP has to provide to its central point all the alerts information

with a true alert list (by human being's analysis if necessary). After all the Val_{TA} values for the training data set instances were calculated using function f_{ASRs} , we then need to tune a threshold for Val_{TA} to decide whether the alert is true attack or false alarm. Based on the configuration of each subscribed IDS, one possible approach is that we could just use the brute-force method to get the value of this threshold, which could generate lowest false positive rate. There could be many useful approaches to come up with the function f_{ASRs} , especially in the statistic and data mining areas, and here we did some preliminary researches and described three techniques.

3.2.1 Brute-Force Method for Weight

We assume f_{ASRs} to be a linear function first.

Definition 8.1: For each alert $a_i(t)$, $f_{ASRs} \{ \} =$

$$\begin{aligned}
 & Constant_{AL} * ASR_{AL_R}(s_i) + \\
 & Constant_{VV} * ASR_{VV_R}(e_i, v_i) + \\
 & Constant_{AM} * ASR_{AM_R}(s_i, m_i) + \\
 & Constant_{AC} * ASR_{AC_R}(s_i) + \\
 & Constant_{VC} * ASR_{VC_R}(e_i) + \\
 & Constant_{AR} * ASR_{AR_R}(s_i) + \\
 & Constant_{VR} * ASR_{VR_R}(e_i) + \\
 & Constant_{VP} * ASR_{VP_R}(e_i, m_i) + \\
 & Constant_{VCC} * ASR_{VCC_R}(s_i, e_i) ;
 \end{aligned}$$

Notation: Constants are tunable, yet must add up to be "1" for normalization; and each ASR_* () value is defined previously.

The way to figure out each of the constant weight value is to use brute-force method, by trying all the possible values for each constant, for example, each constant assigned from 0.01 to 0.90 step by 0.05, provided that they added up to be "1". We could then find the best constant value set among all the training data set for each local IDS, which would provide us statistically highest Val_{TA} for all its true alerts, and lowest Val_{TA} for false alerts. And a threshold could then be tuned in between these values to achieve the best false positive rate with minimum false negatives. This training process has to be done at least once at the beginning, and could also be repeated on regular bases to reflect any changes so as to tune the constants to new values.

3.2.2 One-Rule and Bagging Method

By applying two data mining algorithms "1-Rule" and "Bagging" to the training data, we also could derive function f_{ASRs} , and predict Val_{TA} for the testing data set.

Definition 8.2.1: $\forall a \in A$, training data set $T = T_t + T_f$,

$$threshold_a = \min_{t \in T_t} \{ ASR_a(t), \text{ if } ASR_a(t) > 0 \};$$

$$probability_a = 1 - ER_a;$$

where

$$\text{Error Rate } ER_a = (TN_a + FP_a) / |T|$$

$$\text{True Negative } TN_a = \sum_{v \in T_t} \text{Sum}\{1, \text{ if } ASR_a(t) < threshold_a\};$$

$$\text{False Positive } FP_a = \sum_{v \in T_f} \text{Sum}\{1, \text{ if } ASR_a(t) > threshold_a\};$$

Definition 8.2.2: For each new alert $a_i(t)$, The non-weighted vote

$$f_{ASRs} \{ \} = \sum_{a \in A} \beta(threshold_a, ASR_a(a_i(t))) ;$$

where Function:

$$\beta(probability_a, threshold_a, ASR_a(a_i(t))) =$$

$$\begin{aligned} & \{ 1, \text{ iff } ASR_a(a_i(t)) \geq threshold_a \\ & -1, \text{ iff } ASR_a(a_i(t)) < threshold_a \} \end{aligned}$$

Note: $threshold_a$ defined in 8.2.1

Definition 8.2.3: For each new alert $a_i(t)$, The weighted vote

$$f_{ASRs} \{ \} = \sum_{a \in A} \beta'(probability_a, threshold_a, ASR_a(a_i(t))) ;$$

where Function:

$$\beta'(probability_a, threshold_a, ASR_a(a_i(t))) =$$

$$\{ probability_a, \text{ iff } ASR_a(a_i(t)) \geq threshold_a$$

$$(-1) * \text{probability}_a, \text{ iff } \text{ASR}_a(a_i(t)) < \text{threshold}_a \}$$

Note: probability_a , threshold_a defined in 8.2.1

First, we applied the simplest classification rule called 1R for "1-rule" to each of the ASR knowledge rules. Let's take any ASR_a as an example, each instance in the training data set has this ASR_a of certain value. By nature, we are aware that the each ASR_a values mean some kind of significance of the true attack. With this assumption, we could just break the numeric ASR_a value set into two partitions, the break point value set to be the threshold_a . And since we knew the result of this instance is either true attack or false alarm, we could come out with a classifier that if ASR_a is bigger than the threshold_a , we will concluded that this instance is true attack, otherwise, it was false alarm. We could tune the threshold_a for certain values, and one of the good candidates is the minimum non-zero of ASR_a for the true attack instance. Of course, there would be some error rate when utilizing this classifier. The error rate ER_a defined as those true attack instances recognized as false alarms plus those false alarms recognized as true attacks divided by total number of instances in the training data set. Also, we could define the correctness rate probability_a for ASR_a as $(1 - \text{ER}_a)$. Similarly, we could derive other classifiers for each ASR rule available. Second, we used Bagging aggregation method, to let each of the ASR classifier to vote for the final result. There are two options here, one is non-weighted vote (definition 8.2.2), which each ASR_a count 1 for equally power; another is weighted vote (definition 8.2.3), which we applied $(1 - \text{ER}_a)$ as the probability to be the voting power of each ASR_a classifier either concluded to be true attack or false alarm. After all the ASR classifiers are applied, if the probability sum of true attack is bigger than the probability sum of false alarm, $f_{\text{ASRs}} > 0$, we will predict the testing instance to be true attack, otherwise, $f_{\text{ASRs}} < 0$, it is false alarm.

3.2.3 Native Bayer Method

We could use a statistical modeling called Bayer's rule of conditional probability, which is to use all the attributes and allow them to make contributions to the decision that are equally important and independent of one another, to define function f_{ASRs} as well.

Definition 8.3.1: $\forall a \in A$, training data set $T = T_t + T_f$,

We define two pair of mean and standard deviation values as

$$\text{mean_T}_a, = \sum_{t \in T_t} (\text{ASR}_a(t)) / |T_t|;$$

$$\text{mean_F}_a, = \sum_{t \in T_f} (\text{ASR}_a(t)) / |T_f|;$$

$$\text{std_dev_T}_a = (\sum_{t \in T_t} ((\text{ASR}_a(t) - \text{mean_T}_a)^2 / |T_t|)^{1/2};$$

$$\text{std_dev_F}_a = (\sum_{t \in T_f} ((\text{ASR}_a(t) - \text{mean_F}_a)^2 / |T_f|)^{1/2};$$

Definition 8.3.2: For each new alert $a_i(t)$,

$$f_{ASRs}\{\} = N_t / (N_f + N_t) ;$$

Function:

$$N_t = \prod_{\forall a \in A} N(\text{mean_T}_a, \text{std_dev_T}_a, x_a) ;$$

$$N_f = \prod_{\forall a \in A} N(\text{mean_F}_a, \text{std_dev_F}_a, x_a) ;$$

where $x_a = \text{ASR}_a(a_i(t))$ and function N as the probability density function for a normal distribution with mean m and standard deviation d are given:

$$N(m, d, x) = (2\pi)^{-1/2} d^{-1} e^{-(x-m)^2 / (2dd)}$$

Note: mean_T_a , std_dev_T_a , mean_F_a , std_dev_F_a are each ASR_a related four results defined in 8.3.1

First, we applied the simplest statistic calculation to each of the ASR knowledge rules. Let's again take ASR_a as an example, each instance in the training data set has this ASR_a of certain value. And since we knew the result of this instance is either true attack or false alarm, we could come out with two pairs of mean and standard deviation values for both the true attacks and false alarms in the training data set. So that we could have two normal distribution probability density functions one for true attack prediction, another for false alarm prediction. Second, We applied the value of $\text{ASR}_a(t)$ from a new coming alert instance t to both ASR_a probability density functions and could get two probability density values as true attack probability density and false alarm probability density. By assume that each ASR rule are equally important and independent of one another, We could then apply directly to the native bayer methods, which we multiple each ASR's true attack probability density values together to get N_t , and multiple each ASR's false alarm probability density values together to get N_f . The f_{ASRs} is then defined as $N_t / (N_t + N_f)$; If the final result of $f_{ASRs} > 50\%$, we could predict that the instance is true attack, otherwise, it is false alarm.

4 Prototype Development

4.1 Testing Environment

4.1.1 Background

Testing data and environment was built using data from MIT Lincoln Laboratory DARPA Project 1999. An intrusion detection evaluation test bed was developed which generated normal traffic similar to that on a government site containing 100's of users on 1000's of hosts. In 1999 DAPRA off-line intrusion detection evaluation, more than 200 instance of 58 attack types were launched against victim UNIX and Windows NT hosts in three weeks of training data and two weeks of test data.

Several different teams participated in this project with their most recent developed Intrusion Detection System. The result of each IDS will be a list of detected Alerts, including both true alerts and false alarms.

4.1.2 Simulation

We used each IDS generated alert identification list in 1999 MIT DARPA Project to simulate real-time alert sequence from these IDSes. A process simulated each subscribed IDS, and to read each detected alerts lists, and send the alerts in HDIS (ie. date, method, category, time, duration, source, destination) to DSAP in real time. Time sequence is considered to synchronize different detected alerts from different IDSes. We also implemented all seven hacking behavior analysis algorithms to generate the ASR rules. This process also simulated to push the rules back to local IDS in real time. Since MIT did provide the true alert list of their evaluation test, we can calculate the false positive rate for each IDS based on that.

In the simulation, we have selected total eight IDSes to subscribe. Each of them sent alerts to DSAP, while we were generating the ASR rules based on our correlated hacking behavior algorithms using a simulated DSAP central point process. At the same time, we picked two IDSes to calculate the True Attack Confidence Value (ValTA) of every alert from them using the ASR rules that they received from DSAP at real time. We tested with all the three approaches using different models to evaluate ValTA. The results were then used to report the false positive rate. Due to the length of the paper, we just presented the results using data from Audit Data Analysis and Mining System from Center of Secure Information Systems, George Mason University.

4.2 Testing Results

In order to get meaningful results, we break the input data into two time range, using data from 3/29 - 4/7 as training, and find the most proper parameters, and then use these values directly testing the data from 4/8 - 4/9. Look at both false positive rate, and missed true alert number, you would get the general picture about the dilemma between false positives and false negatives. This way to simulate how the system supposed to be applied.

4.2.1 Brute-Force for Weight

- Training Result (03/29/1999 - 04/07/1999)

ASR	AL	VV	AM	AC	VC	AR	VR	VP	VCC
weight	.005	.005	.005	.005	.005	.005	.005	.325	.640

- Testing Result (04/08/1999 - 04/09/1999)

Testing Result (4/8-4/9)	Threshold (Val _{TA})	Detected True Alert	Missed True Alert	False Alarm	Filtered False Alarm	False Alarm Rate
Original Result	N/A	7	0	6	0	46.2%
New Result	0.328	6	1	2	4	25%

4.2.2 One-Rule and Bagging Method

- Training Result (03/29/1999 - 04/07/1999)

ASR	AL	VV	AM	AC	VC	AR	VR	VP	VCC
Proba- bility	.691	.527	.527	.527	.745	.600	.691	.255	.727
Thres- hold	.028	.005	.059	.022	.005	.059	.027	1.000	1.000

- Testing Result (04/08/1999 - 04/09/1999)

Testing Result (4/8-4/9)	Threshold (Val _{TA})	Detected True Alert	Missed True Alert	False Alarm	Filtered False Alarm	False Alarm Rate
Original Result	N/A	7	0	6	0	46.2%
Weighted Vote	0.0	6	1	1	5	14.3%
Non- Weighted	-1.0	6	1	2	4	25.0%

4.2.3 Native Bayer Method

- Training Result (03/29/1999 - 04/07/1999)

ASR	AL	VV	AM	AC	VC	AR	VR	VP	VCC
True Attack Mean	.168	.039	.284	.056	.066	.477	.356	0	.756
True Attack Standard Deviation	.202	.071	.382	.107	.083	.447	.253	0	.429
False Alarm Mean	.075	.200	.245	.027	.068	.406	.435	0	.357
False Alarm Standard Deviation	.101	.348	.409	.053	.115	.458	.377	0	.479

- Testing Result (04/08/1999 - 04/09/1999)

Testing Result (4/8-4/9)	Threshold (Val _{TA})	Detected True Alert	Missed True Alert	False Alarm	Filtered False Alarm	False Alarm Rate
Original Result	N/A	7	0	6	0	46.2%
New Result	50%	3	4	2	4	40%

note, if certain ASR's standard deviation is zero, that ASR will be ignored.

4.3 Conclusion and Future Research

From the testing results, we could conclude that by using our correlated algorithms to measure the Val_{TA} of each coming detected alert, it could help the security administrator to set priority for those alerts for further processing. If certain threshold applied, lower false positive rate and less alert reporting instances could be achieved at the cost of missing some true alerts as well. We also found that some approaches to evaluate the ASR values are better than others, yet, due to the limit amount of training and testing data so far, some advantages of the algorithm may still need to be explored.

We realized that the data set we tested for now is very insufficient, because the submitted testing results has only hundreds of alerts within two weeks, and there are not many established hacking behaviors embedded in that testing bed environment. We used this simulation just to show the concept, and we would like to test more real hacking data with different IDS resources in our future research, especially those IDS testing bed and benchmark sources. With more testing on those data from real hacking scenario, we'll expect to see more significant results.

Future research will study more correlated algorithms to enlarge the knowledge base, which could then be used in broader IDS facets other than false positive measurement and reduction. And implementation of the communication with consideration regarding scalability and performance on computational expense will also be further developed.

References

1. Steven Hofmeyr, Stephanie Forrest, Patrik D'haeseleer "Distributed Network Intrusion Detection, An Immunological Approach", Dept. of Computer Science, University of New Mexico, May 1999
2. Herve Debar, Ming-Yuh Huang, David J. Donahoo "Intrusion Detection Exchange Format Data Model", Internet Engineering Task Force, IDWG, IBM Corp./The Boeing Company/AFIWC, March 2000
3. "Intrusion Alert Protocol - IAP", Internet Engineering Task Force, IDWG, Gupta, Hewlett-Packard, March 2000
4. Julia Allen, Alan Christie, William Fithen, John McHugh, Jed Pickel, Ed Stoner, "State of the Practice of Intrusion Detection Technologies", Carnegie Mellon Software Engineering Insititute, Jan 2000
5. Richard Lippmann, Joshua W. Haines, David J. Fried, Jonathan Korba, Kumar Das, "The 1999 DARPA Off-Line Intrusion Detection Evaluation", Lincoln Laboratory MIT, 2000
6. Ming-Yuh Huang, Thomas M. Wicks, "A Large-Scale Distributed Intrusion Detection Framework Baed on Attack Strategy Analysis", The Boeing Company,
7. Ian H. Witten, Eibe Frank, "Data Mining, Practical Machine Learning Tools and Techniques with Java Implementations", University of Waikato, 2000. Morgan Kaufmann Publishers.
8. Y. Frank Jou, Shyhtsun Felix Wu, Fengmin Gong, W. Rance Cleaveland, Chandru Sargor, "Architecture Design of a Scalable Intrusion Detection System for the Emerging Network Infrastructure", North Carolina State University, Apr 1997

9. Peter Mell, "Understanding the World of your Enemy with I-CAT (Internet-Categorization of Attacks Toolkit)", NIST, Computer Security Division, May 1999
10. Robert A. Clyde, Drew Williams, "Enterprise-wide Intrusion Detection, A multi-tiered approach", AXENT Technologies, Inc, 1998
11. Peter G. Neumann and Phillip A. Porras, "Experience with EMERALD o Date", SRI International, 1999
12. Steven Cheung, Rick Crawford, Mark Dilger, Jeremy Frank, Jim Hoagland, Karl Levitt, Jeff Rowe, Stuart G. Staniford-Chen, Raymond Yip, Dan Zerkle, "The Design of GrIDS: A Graph-Based Intrusion Detection System", Jan 1999
13. Wenke Lee and Sal Stolfo. A Framework for Constructing Features and Models for Intrusion Detection Systems. ACM Transactions on Information and System Security, November 2000
14. Tim Bass, "Multisensor Data Fusion for Next Generation Distributed Intrusion Detection Systems", Silk Road & ERIM International, 1999
15. Salvatore J. Stolfo, Wei Fan, Wenke Lee, "Cost-based Modeling for Fraud and Intrusion Detection: Results from the JAM Project", Columbia University, 1999

Design of UNIX System for the Prevention of Damage Propagation by Intrusion and Its Implementation Based on 4.4BSD

Kenji Masui, Masahiko Tomoishi, and Naoki Yonezaki

Department of Computer Science,
Graduate School of Information Science and Engineering,
Tokyo Institute of Technology.
2-12-1 Oookayama, Meguro-ku, Tokyo, 152-8552, Japan.
{masui,tomoishi,yonezaki}@fmx.cs.titech.ac.jp

Abstract. On usual UNIX systems, a privileged user of root is allowed to acquire any user's authority without authentication process. If an intruder obtains the root privilege by taking advantage of system's security hole, he can abuse network reachability of any user of the system to break into other sites. Thus we present a new system design where the authority of users is protected from root by introducing a new user substitution mechanism. However, even if we introduce the new mechanism, on usual UNIX systems, the intruder can get the authority using many other methods for root. We implement the new user substitution mechanism and the mechanisms which prevent the intruder from using such methods in FreeBSD-4.2, and confirm that the system design is effective.

1 Introduction

As the Internet grows, the number of users on the network is rapidly increasing. With the growth of the network, the number of illegal accesses are also increasing. In recent years, not only independent applications such as e-mail or web browser but also combined services like electronic commerce are available. Therefore illegal access may cause economic damage as well.

In order to devise a countermeasure against illegal access, many aspects of network security have been studied actively. Studies on cryptographic theory[1] and verification of communication protocol[2] provide various ways to protect systems from illegal access through network. Theoretically speaking, by using formal proof methods[3][4][5], we can prove that a program does not contain bugs of some types, and that a intrusion by means of taking advantage of the bugs does not occur. However, it costs a lot to encrypt or verify all programs and data on a system because there are a large number of programs and data on a practical system. In addition, there are many "legacy programs" that although are no longer maintained, we have to use them. Therefore it is hard to ensure safety of all programs on a system.

System protection is one of the main issues in security related studies. Most of the research aim to prevent intruders from breaking into systems. However, as we

described above, it is highly improbable to ensure safety of programs completely. Hence we have to find a mechanism which prevents the damage of intrusion from spreading when an intruder takes advantage of a flaw in a program and breaks into the system. In this paper, we propose such a mechanism.

There are many UNIX systems on the Internet and they bring us various services. In order to achieve our aim stated above, we extend a UNIX system. On usual UNIX systems, a privileged user of root has a very strong authority. A process running at root authority is allowed to transfer its authority to that of any user unconditionally by using *setuid* system call, and to read and write all files regardless of the state of permission bits. On the other hand, when a process of general user requests to have other user's authority, password authentication is required. That is, there is absolutism of root on usual UNIX systems. In the literature [6], root privilege is described in detail.

A UNIX system is widely accepted as being easy to administrate because root privilege of UNIX makes it easy and intuitive to handle authority of users. However, a system construction method which depends on root privilege may cause a problem in network environment. Many network service programs of UNIX are executed at root authority in order to get access right to resources needed for services. In many cases of intrusion, an intruder takes advantage of security holes of network service programs and breaks into a system. Usually, root authority is closed within the system, while general users may have corresponding accounts on other systems and they may reach the system by means of network protocol such as rlogin. In short, root is classified as local user, while general users are classified as network user. These classifications are described in detail in section 2.2. In the case of intrusion, the intruder can not only destroy the system he first got in, but also attack other systems by changing his authority from root to any other user on the system and abusing the user's authority. If the damage of intrusion reach other systems or organizations, the organization which allowed intrusion in first lose its trust as well as resources on the systems.

To solve the problem, we propose a new system which protects authority of a network user when an intruder obtains root privilege. In section 2, we analyze a process for intruder abusing of network users' authority, and we propose some mechanisms as countermeasures for each step of the process. From section 3 to 7, we describe individual mechanism proposed in section 2 in detail. In section 10, we compare our system with some systems proposed in other researches that take approaches similar to ours. In section 11, we describe the implementation of our system.

2 Policy of Protection

2.1 An Intrusion into a UNIX System

There are many known tricks to intrude into a UNIX system. In literature [6], many ways of intrusion such as, using default accounts of a system, guessing passwords, monitoring network packets, using a secret path called "backdoor",

and taking advantage of security holes of network daemon programs, are described. Among them, we regard the way exploiting security holes of network daemon programs as serious.

The first reason is that the attacks in this way is occurred frequently. According to “CERT/CC Advisories” [7] investigated and reported by CERT¹, a lot of instances of illegal access are reported. Most of the illegal accesses mentioned in the report are by means of exploiting security weakness of network daemons. There are many different kinds of network daemons, and a weakness of security might be generated by a subtle mistake of programming. Generally, it is hard to examine whether a weakness exists in a program. Moreover, if such a weakness is found in a legacy program, it might be hard to fix the program. Therefore we hardly can investigate all programs to confirm the existence of weaknesses.

The second reason is that it is possible to prevent intrusions using other ways, by the efforts on administration as follows.

- Disable default accounts.
- Use passwords hard to guess.
- Do not send password over the Internet.
- Replace a program of which existence of backdoor is known.

2.2 Network Users

Since we adopt the assumption that a system may be intruded, it is important to minimize the damage caused by an intruder. In this section, we define a class of users who should be protected.

There are two classes of users on UNIX systems. One is the class which contains administrative users such as root and nobody. The other is the class which contains general users. The user of root exists on almost all kinds of UNIX systems. But generally, there is no correspondence between root on a host and root on the other host though they have the same name “root”. The same argument holds for “nobody”. That is, root and nobody are “local users” within a host, while general users are the accounts for people in real life, and there may be corresponding accounts on other hosts over the network. In this sense, general users are “network users”. Network users often have “network reachability”. For example, a user who has network reachability is allowed to do remote login without passwords, or he can access files on remote host via NFS. In short, we say that a user has network reachability if he is allowed to access resources on other hosts over the network without password authentication.

At present, most of the systems are set up so as not to give network reachability to local users. On the other hand, network users often have network reachability, and there are some risk of their authority being abused by intruders. For example, in case an intruder obtains the authority of network user “Alice”, the intruder can alter all files which Alice is allowed to access via NFS, as well as files on the local system. Moreover, the intruder is able to login to all hosts to which Alice can login by rlogin protocol.

¹ Computer Emergency Response Team.

Therefore in this paper, we want to construct a system such that it is hard for an intruder to get authority of network users on the system even if he obtains root authority of the system. With this system, we can stop spreading the damage of intrusion, as the intruder attacks other hosts or sites. The attack is called “Island hop”.

2.3 A Way to Weaken Root Privilege

We weaken some privileges of root, in order that intruders cannot abuse authority of network users. There are 2 security layers in UNIX systems. One is root privilege, and another is non-root authority. In some OSs but UNIX, there is a multi-level security layer such as ring protection mechanism in Multics[8]. In this paper, we construct a new security layer, which is weak root authority layer, to protect authority of network users from intruders. Possible attacks of an intruder used in the process, from obtaining root authority to abusing users’ network reachability, are shown in figure 1. Each node in the figure represents a attack which the intruder is able to do. If the node “dispatch from network port” is reachable from the node “take advantage of daemon’s security hole”, intruder will succeed to abuse authority of network users. Since we have adopted the premise that an intrusion will occur by exploiting daemon’s security holes and the intruder executes processes of root authority, it is impossible to prevent operations of node 1 and 2 in this setting. It is also impossible to prevent the operation of the node 7 because it is an usual login procedure. Neither the flow of node from 8 to 10 can be prevented because it is usual network access by users. Usability of the system would be reduced if we prevented the flow. Therefore we put restrictions on operations in the nodes in figure 1 except the nodes mentioned above, so that it is impossible to reach node 10 from node 1. Note that the way of stealing passwords by monitoring network or installing the Trojan horses, and that of logging into remote hosts by means of the passwords which they stole, are not included in figure 1 because accessing remote hosts by password authentication is not a kind of attack by network reachability that we defined in section 2.2

In order to prevent intruders from getting control of network users’ process, first we add password authentication to user substitution mechanism of UNIX. We describe the details of the mechanism in section 3.1. By introducing the mechanism, we restrict the operation of node 5 so that only a person who knows the correct password of a user can get the authority of that user. Though we introduce the restriction, such that the intruder can execute a process of general users’ authority. This operation corresponds to the flow 3→4→8 in figure 1, because there is “suid bit” on UNIX systems which specifies authority of the process on execution and root is allowed to modify configuration of the suid bit arbitrarily. To prevent this, in section 3.2, we put restriction on the operation of node 4 and prevent intruders from obtaining authority of general users by means of suid bit. In the same section, we show that it is better to restrict the operation of node 4 than node 3, although it is also possible to restrict the operation of node 3. In addition, since root can read and write all files on the system, an intruder

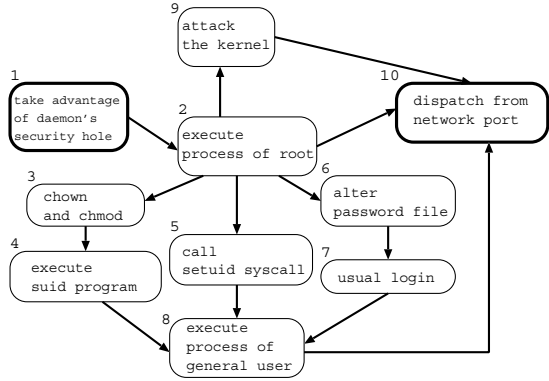


Fig. 1. The attack for the abuse of network users' authority

is able to get authority of general user by proper authentication procedure if password files are replaced with the passwords he created. The flow 6→7→8 in figure 1 corresponds to the attack. To restrict the operation of node 6, in section 4.1 we propose a new mechanism which protects the password file from altering by root. As a seed of another possibility, root has the privilege of reading and writing the memory inside the kernel. In this case, there is the possibility that operation which should be denied by kernel can be executed by root. If it is possible for an intruder to control the kernel, any operation can be executed and authority of users can be stolen. To prevent the operation of node 9 in figure 1, we take the privilege to access in-kernel memory away from root in section 4.2. In addition, in section 7 we prevent intruders from replacing the kernel itself. Remaining possible attack is by using the privilege to dispatch from arbitrary network ports on the system. This flow is 2→10 in figure 1. In this case, even if the intruder has not obtained authority of network user, he is able to cheat remote systems into believing that he has the authority of the user by using some protocols such as `rlogin`. To prevent this, we propose a new mechanism in section 5. The mechanism allows only specific programs authorized by the kernel to use privileged network ports of the system.

Readers may wonder about figure 1. You may want to ask “Is there other ways for root to obtain network user’s authority?” It is difficult to answer the question. We have the same question about protection of root authority from other (general) users. They say usual UNIX system’s root authority is well protected. i.e. “No general user can gain root authority without authentication.” However, to prove that assertion is very difficult. The large number of functions and facilities must be checked to prove that assertion, and if they miss one security hole the proof is incorrect. In spite of that situation, we do not think our system is useless. The reasons are:

- When we find missed functions or facilities, we can disable it or modify it to suit our protection scheme.
- We can realize our protection mechanisms as components of structure of security layers.

We introduce security layers into UNIX systems. Usual UNIX systems have single-user mode and multi-user mode. These two modes seems to belong distinct layers. But it is not true. Since a root in multi-user mode has several hardware level access rights, he also has authority of a root in single-user mode. In order to divide these two root's authority completely, we introduce new mechanisms mentioned above. Therefore, if there is particular function which gives single-user level access rights to multi-user mode's root, we realize that the function must be disabled or modified to serve its function in controlled way.

3 Introduction of Restriction on User Substitution

3.1 Introduction of RSU

To prevent an intruder from invoking *setuid* system call, we introduce a new right RSU². RSU is a set of user IDs and is attached to each process. Elements of RSU can be added and removed during the execution of a process. Each process can transfer its authority to a user ID included in the RSU of this process by calling *setuid* system call. When a process calls *setuid* with the parameter of destination user ID, the kernel checks the RSU of the caller process and changes the authority of the process if the destination user ID is in the RSU. We say “A process has RSU of target user” when the destination user ID is in the RSU of the process. A process can add a user ID to its RSU if the process is authenticated using a password by means of *rsu* system call newly-established. The behaviour of *rsu* system call is as follows.

System call *rsu(uid,pass)*.

A process calls *rsu* with *uid* and *pass* as parameters. The user ID *uid* is added to RSU of the caller and the kernel returns successfully, if the password *pass* is correct. A call fails if the password *pass* is wrong.

RSU is inherited to the child process created by calling *fork* or the process replaced its core-image with an executable file by calling *exec*. The process *init* which is a parent of all processes starts its execution with empty RSU.

To call *setuid* successfully, a process needs to have RSU of its target user. To get RSU of target user, a process needs to know the password of the user.

Since network daemons are running with no RSU of users, an intruder cannot obtain authority of users unless he knows their passwords.

² Right of Substitute User ID

3.2 Suid Bit

“Setuid on exec bit” (we call it “suid bit” from now on) of UNIX file-system is another mechanism to change authority of a process. Usually, a process runs at authority of a user who started the process. When a suid bit of an executable file is set, the process runs at the authority of the owner of the executable file regardless of the authority of the executor. With this mechanism, an intruder can execute any executable file at any user’s authority without *setuid* system call, because root is allowed to modify suid bit and owner ID of any file. This attack corresponds to the flow $2 \rightarrow 3 \rightarrow 4 \rightarrow 8$ in figure 1. Below we show two ways to prevent this attack.

1. To prevent intruders from creating a program file which has the states where its owner is general user and its suid bit is set, we restrict the operation of node 3 in figure 1. We change the behaviour of *chown* system call so as to permit a process to do *chown* when the process has RSU of the target user. In addition, we modify *chmod* system call that changes access right³ of files so that the system prohibits setting suid bit of a file owned by the other user.
2. To limit user IDs for which suid bit is valid on execution, we restrict the operation of node 4 in figure 1. We make a suid bit of a file valid if the file is owned by local users. e.g. root, news and uucp. User IDs for which suid bit is effective can be added or removed by editing the configuration file.

In the case 1, in addition to the modification of behaviour of the system calls, we must prevent root from accessing file-systems via raw devices in multi-user mode. If root can access raw devices, all the data on a file-system such as the contents of the files, access right bits or owner ID of the files might be altered. Since we modify both *chown* and *chmod* system calls, unusual work may be needed to administer the system and more burden⁴ may be imposed on administrators. In the case 2, it is enough to modify only *exec* system call. Moreover, there is little influence on system administration because the program file which has the state where its owner is general user and its suid bit is set hardly exist on systems in real scenes. Since the number of changes on the system is smaller and the system is easier to administer than the former method, we believe that the latter method is better and adopted the latter method in this research.

4 Protection of Information for Authentication

Root can alter */etc/passwd* and replace users’ passwords by known passwords because root is almighty about file-systems. This attack corresponds to the op-

³ It includes some bits such as permission bits of read/write and suid bit

⁴ When a new user’s home directory is created, extra process is required such as setting temporary password of the user because it is impossible on this system to *chown* without RSU.

eration of node 6 in figure 1. In this section, we propose a new method to protect information and mechanisms used for authentication from being altered by root.

4.1 Protection of Configuration Files

We can think of many possible ways to protect files from intruders. In this research, we introduce a new file-system that root cannot write into it and we keep the files in the file-system. Device drivers for accessing disks are protected from root because these drivers are inside the kernel. In addition, this method is safe since it does not depend on other hosts or external programs. From now on, we call this new file-system weeded-fs.

weeded-fs must satisfy the following conditions: 1) cannot be accessed by root. 2) access to raw devices is prevented.

4.2 Protection of Information Inside the Kernel

A usual UNIX system has device files as interfaces to export information inside the kernel. These interfaces have the names `/dev/kmem` and `/dev/mem`. There is a danger that the control of the kernel may be captured by an intruder, if a root process under control of the intruder can read and write these files. This attack corresponds to the operation of node 9 in figure 1. To prevent this, we prohibit root from accessing these files like weeded-fs. In this case, we can use process file-system or sysctl interface[9] instead of `/dev/kmem` and `/dev/mem`.

5 Programs Authenticated by the Kernel

It is possible for an intruder to login to remote hosts as a network user without getting the user's authority, if the process of the intruder is able to dispatch from privileged ports to network[5]. This attack corresponds to the flow from 2→10 in figure 1. The intruder succeeds in abusing authority of network users by means of `rlogin` command. To keep this attack away, we have to prevent intruders from using usual `rlogin` command and replace it with a new version which checks whether the RSU of the parent process[6] includes the target user ID. There are many network daemons which need to use privileged ports for their services. Such daemon programs must be assured that they have never altered by an intruder. Therefore we need a new mechanism to permit only particular programs which the kernel grants to access privileged ports.

To actualize this rule, we modified the kernel to perform authorization of executable files. System administrators prepare MD5[10] hash values of executable files at installation time and store these values into the weeded-fs. When a program is executed, the kernel calculates MD5 hash value of the program. Access rights of privileged ports or some protected resources are given to the process if

⁵ Due to the specification of rlogin protocol.

⁶ This process is a user's shell in most cases.

the calculated value is equal to the stored value on the weeded-fs. We can prevent root from altering a program because it is hard to alter a program with holding the same MD5 value for the characteristic of MD5 algorithm. Programs such as accessing below the IP layer of network or using privileged ports of network must be authenticated by the kernel. SecureBSD [11] has similar mechanism. Our advantage against SecureBSD is that we have weeded-fs. Since weeded-fs is accessible by authenticated users except root in multi-user mode, an administrator can install new MD5 value in weeded-fs and the MD5 value is protected from root.

We do not recommend to use `rlogin`. It is only a typical example of a command which requires MD5 checking. `rlogin` system should be replaced by `ssh` or other secure networking tools. In that case, you may want a mechanism which transmits RSU information from a kernel to another kernel. We think that mechanism is feasible as follows:

1. Each host (kernel) has a public/private key pair.
2. A network client (say `ssh`) asks the kernel to sign a message and sends the signed message to remote server.
3. Remote server receives the message and asks the (remote) kernel to validate it.
4. If the message is valid, the (remote) kernel regards the client process as authenticated process, and allows the server to create user's process.

6 Compatibility with Usual Authentication Method

When a process provides correct password to the kernel, the RSU of the user who corresponds to the password should be given to the process. But usual programs never call `rsu` system call since they are created without the concept of RSU. On the system that we propose in this paper, since a process without RSU of target user cannot change its authority to the user, basic authentication commands such as `login` which change their authority by calling `setuid` system call may have trouble.

To make these programs compatible with our system, we modify library functions such as `getpwent()` and `crypt()`, and we add a new facility of getting RSU by calling the kernel to these functions.

6.1 Modification of Password Functions

We modify the following library functions. On usual UNIX systems, these functions behave as follows.

Password search function *getpwnam(name)*.

This function returns a record of information of account specified by *name* parameter. The record includes user ID, hashed password and so forth.

Password hash function *crypt(key,salt)*.

Using a password given as *key* and a parameter *salt*, the function calculates hashed password and returns it.

On our system, we modify these functions⁷ as follows.

- A process can get password information only via system call because the password file is protected by the kernel. For that reason, we introduce a new *getpwnam* system call. The library function *getpwnam()* calls this system call, gets password information and returns it.
- The library function *crypt()* passes the password given as a parameter to *rsu* system call. If the password is correct, the user ID which corresponds to the password is added to the RSU of the caller by the effect of *rsu* system call. Subsequently, *crypt()* calculates hashed password as usual and returns it.

By the modifications mentioned above, when a process passes a correct password to *crypt()*, the RSU of the user who corresponds to the password is given to the process. After that, the process can transfer its authority to that of the user.

Readers may wonder if modifications of these functions are sufficient. There are many commands like “ls -l” that requires a translation from a user id to login name. Do these commands work properly? The answer is yes. Such translation routines call *getpwuid()* internally and never access */etc/passwd* directly. Most UNIX implementations adopt that way, for uniformed treatment for NIS and NIS+.

7 Protection of the Kernel

By adopting the mechanisms mentioned so far, the authentication system is under the protection of the kernel. However, it is still possible for an intruder to replace the kernel itself with the kernel which he created and reboot the system. This attack corresponds to the operation of node 9 in figure 1. 4.4BSD⁹ proposed a solution for this problem. There is a variable of “securelevel” in the kernel of 4.4BSD. It is impossible to turn off “immutable” flag on file-systems while the variable is set to 1, and it is impossible to modify or rename the file of which immutable flag is set.

Since the value of *securelevel* cannot be decreased unless the system is shut-down, we can prevent an intruder from replacing the kernel while the system is in multi-user mode if immutable flag of the kernel file is set.

However, the *securelevel* mechanism is incomplete to protect the kernel from an intruder who has root authority. We show the state transition diagram of *init* program of 4.4BSD in figure 2.

“runcom” is the state where *init* is executing */etc/rc*⁸. *Securelevel* is set to 0 at single-user mode and set to 1 in multi-user mode. According to figure 2, after the state of *init* changes from single-user mode to runcom, *securelevel* will be set to 0. If an intruder creates a script which replaces the kernel in */etc/rc* and reboots the system, */etc/rc* may run while *securelevel* is set to 0 and the

⁷ Functions like *getpwent()* must be modified as well.

⁸ A script executed at startup of the system.

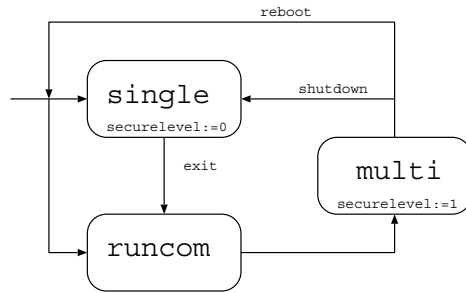


Fig. 2. The state transition diagram of init program

kernel may be replaced. To prevent this attack, there are two simple ways: 1) Store `/etc/rc` script in weeded-fs. 2) Set `securelevel` to 1 when `init` enters `runcom` state. However, the former will lose flexibility of administration and the latter will arise a problem when some programs such as `fsck`⁹ are invoked. For the solution, we combine these two ways. We install a script which should be executed while `securelevel` is 0 in weeded-fs, and a script which should be executed while `securelevel` is 1 in usual file-system.

In this case, the intruder's script does not run automatically while `securelevel` is 0. Therefore the kernel cannot be replaced in multi-user mode. In addition, administrators can edit such a startup script in usual file-system as it simply launches some daemons.

8 Treatments for Non-interactive Programs

There are some programs such as `cron` and `sendmail` that create child processes which run at authority of general users. On our system, that kind of program cannot run since the programs are not interactive and not able to get RSU while the process is running. On the other hand, interactive programs such as `login` and `su` are able to get passwords and RSU while they are running.

To save `cron` and `sendmail`, we add a new functionality to our system. We already introduced MD5 checking on `exec` system call. We can use similar approach to give a process extra RSU information. When `cron` like program is executed, the kernel checks its MD5 value and compare with registered value. If these two values are equal, the kernel gives the process extra RSU.

This idea is good and bad. For `cron`, we worry about contents of `crontab`. If intruders can edit user's `crontab`, they can abuse user's authority. However, we can solve this problem by placing `crontabs` on weeded-fs. For `sendmail`, to use the function is more dangerous than the case of `cron`. Intruders may exploit security holes of `sendmail`, for `sendmail` often runs as a network daemon. If we give `sendmail` extra RSU, the intruders gets network users' authority. Although

⁹ It checks and repairs consistency in a file-system.

sendmail is developed for long time and well checked than other new software, we cannot assure its perfection. We have to consider and be conscious of tradeoff between convenience and security when using this function.

9 Summary of Modifications

From section 3 to previous section, we described some mechanisms which implement our policies. In this section, we recall modifications that we proposed by stating capabilities of root on our system.

The operations which root cannot do on our system are listed below.

- To get a network user's authority by calling *setuid* system call without RSU of the target user.
 - We modified *setuid* system call to honor RSU information of the caller process. A process can obtain RSU of a user by providing a correct password of the user to the kernel.
- To run programs at a network user's authority by calling *exec* system call on suid programs.
 - We limited the list of the user who are valid as a owner of suid programs. This limit is effective on all file-systems.
- To access files on weeded-fs.
 - For root, weeded-fs is untouchable.
- To access in-kernel memory, physical memory and physical I/O devices.
 - If root could access these devices, he could disable our policies since we implemented the mechanisms as the kernel code.
- To use privileged ports without MD5 authentication by the kernel.
 - Only unaltered programs can use privilege ports.
- To edit a startup script placed in weeded-fs.
 - Root cannot get reboot-attack for the sake of this limitation.

These modifications may introduce some drawbacks.

- Since root cannot execute any process at authority of network user, some programs such as **sendmail** may not work correctly.
- Suid programs owned by network users do not work.
- An administrator cannot access physical memory and I/O devices in multi-user mode. Some type of operations such as formatting disks and testing hardware devices can be done only while the system is in single-user mode.
- All files on weeded-fs are protected while the system is running at multi-user mode. To edit files on weeded-fs, an administrator have to create a non-root user and give him an access permission to the files.
- To install and start new daemon programs, an administrator must calculate MD5 values of the programs and store the values into a configuration file kept on weeded-fs. This procedure is extra burden that did not exist on usual UNIX systems.

10 Comparison with Other Systems

In this paper, we proposed a new method which weakens part of the root privilege to protect authority of network users from an intruder and keep compatibility with existing programs. In this section, we compare our system with other systems. We choose 4.4BSD and Plan9^[12] as targets of comparison. These systems have similar approach to ours, for they put restrictions on root privilege. We show that our system is more flexible than these systems.

Besides those two systems, there are many secure operating systems like CMW+^[13] or Trusted Solaris^[14]. We did not choose them as targets of comparison since they are classified as class B system of TCSEC^[15]. All class B system adopts mandatory access control(MAC)^[16], while usual UNIX systems adopt discretionary access control(DAC)^[17]. We do not think class B systems are suitable as targets of comparison because we did not implement MAC on our system. In addition, We want to keep the influence of modification as little as possible. We made many changes on functions of usual UNIX systems, however, the modified system is still UNIX-like system for users and administrators. On the other hand, introduction of MAC into usual UNIX systems causes drastic change. Therefore, our goal is very different from that of class B systems. Which is also a reason for the choice we have made.

10.1 Security Level Mechanism on 4.4BSD

As we described in section 7, we can protect files from altering in multi-user mode by setting immutable flag of the files on 4.4BSD systems. Therefore we can protect a password file or programs by using this flag. However, the mechanism of immutable flag and securelevel lacks flexibility because immutable flags cannot be changed until the system is shut-down. If we adopt the mechanism as it is, we must shut-down the system whenever we have to edit protected files. On the other hand, it is possible to change passwords or renew MD5 values on our system since users except local users can edit all files on the weeded-fs while the system is in multi-user mode.

10.2 Plan9

Plan9 operating system was developed at Bell Labs^[10] of AT&T. The most significant characteristic of the Plan9 is that it almost abolished root privilege. The special user^[11] which corresponds to root on usual UNIX systems still exists on Plan9. However, that user does not have the privilege that root on a usual UNIX system has.

Plan9 system consists of CPU servers, file servers, authentication servers and terminals. They are connected to each other via the network.

¹⁰ Bell Labs belong to Lucent Technologies now.

¹¹ The user is called “administrative user”.

The most significant security feature of Plan9 is that CPU server and file server are independent machines. That is, CPU server is a different machine from other servers. In case an intruder breaks into a CPU server, the intruder cannot use resources on file servers because the administrative users on CPU server and file server are different. In addition, password files are protected from an intruder because the authentication server is also a different machine from other servers.

On a Plan9 system, a process must be authenticated by an authentication server by means of authentication protocol of Plan9 when the process requests to access resources which is owned by other users. This authentication protocol is based on kerberos [18].

As described above, on Plan9 systems, user substitution is controlled by using authentication. Password files and users' files are protected from intruders by making each server independent of the others.

The difference between our approach and Plan9 is tolerance to intrusion. Even if the CPU server is intruded, password files on the authentication server are safe, because Plan9 adopts distributed structure of servers. However, if an intruder breaks into the authentication server, the entire system is got into danger. In addition, our system is able to work as a stand-alone host, whereas Plan9 cannot work as a stand-alone host without losing security. It is possible to put all services together on a CPU server. In that case, however, the entire system becomes insecure because password files are not protected.

10.3 Result of Comparison

The result of comparison between our system with other systems is shown in table 1. The column labeled "setuid" shows characteristics of user substitution mechanism of each system. "UL" means that user substitution is unlimited. "auth" means that the system has authentication mechanism for user substitution. The column labeled "Prot" shows whether the system has a restriction on accessing resources by root (or special users on that system). The column labeled "Flex" shows whether the system allows users to access protected resources while security mechanisms of the system are enabled. The column labeled "stand-alone" shows whether the system is able to work as a stand-alone host while security protection is enabled.

On our system, password authentication is introduced to the user substitution mechanism, whereas *setuid* system call is not restricted on usual UNIX

Table 1. Comparison with other systems

system	setuid	Prot	Flex	stand-alone
UNIX	UL	no	yes	yes
4.BSD	UL	yes	no	yes
plan9	auth	yes	yes	no
RSU	auth	yes	yes	yes

and 4.4BSD. Our system also has mechanisms such as `weeded-fs` and `securelevel` (which is imported from 4.4BSD system) to protect password files and authentication mechanism itself from altering by an intruder. Usability of our system is almost the same as usual UNIX systems because users except local users are allowed to access protected data while security mechanism is active. Our system can operate as independent host, while Plan9 has no tolerance to intrusion when it is configured as a stand-alone host. Therefore our system controls user substitution by password authentication and has flexibility for administration and can work as a stand-alone host.

10.4 Relations to Chroot and Jail

Many UNIX systems have `chroot()` system call. It restricts namespace of a filesystem to under the specified directory. FreeBSD-4.0 or later have similar `jail()` system call. It creates a virtual host environment and imprisons a process and future descendants. A process cannot release the effects of both system call even if the process runs at root authority. These system calls are intended to protect files from intruders. However, our aim is to stop “Island hop” attacks, not to protect users’ files. Therefore, we cannot compare these system calls with our mechanisms directly. For example, `chroot()` system call does not prevent network access.

11 Implementation

We first implemented proposed system as a patch to FreeBSD-2.2.8, and ported to FreeBSD-4.2 later.

We confirmed that root cannot obtain a process of a network user by usual ways such as invoking `su` command, executing `suid` programs owned by network users, editing the password file and so on. On the other hand, root can invoke `su nobody`, `su news` and `su uucp`. The `suid` bits of files owned by local users are valid. All files on usual file-systems can be handled freely by root. Root can work nearly as usual.

We considered the implementation in detail:

Overheads caused by the modifications. Overheads needed for checking RSU are not so much. `setuid` system call checks whether RSU of the caller includes the uid which is specified as the parameter of the system call. It costs at most the length of RSU set to check RSU set. Overheads for checking RSU can almost be ignored because the size of RSU set is much smaller than the number of users on the system. On the other hand, the cost for calculating MD5 values of programs may not be ignored. As concerns this, we should consider not only the cost for calculating MD5 value but also frequency of MD5 authentication occurrence.

Inter-operability with existing OS. On our system, a login service by `rlogind` does not work because a process without RSU of target user cannot transfer its authority to that of the user. The main reason of this is

that `rlogind` cannot execute user's shell since no password is sent over the network in rlogin protocol and the daemon cannot get RSU of users. On the other hand, it is possible to mount remote file-systems via NFS because the operation of NFS is closed inside the kernel.

Our implementation runs quite fine. FreeBSD-2.2.8 based system ran almost for a year with no trouble. Of course, the system is sometimes rebooted due to administrative reasons. However, it can run for many months continuously. We expect FreeBSD-4.2 based system is also stable.

12 Conclusion

In this paper, we proposed a new UNIX system design on which authority of network users cannot be abused by an intruder if the intruder obtains root authority. There is less danger so-called "island hop" attack than an intruder gets root authority and attacks remote hosts by using authority of network users. In addition, our system design is more flexible than other ones on which root authority are restricted for security. The systems by our design distinguish between network users and local users, and the former can access some resources which the latter cannot access.

References

1. Dorothy E. R. Denning. *Cryptography and Data Security*. Addison-Wesley, 1983.
2. Michael Burrows, Martin Abadi, and Roger M. Needham. A logic of authentication. In *ACM Transactions on Computer Systems*, volume 8, pages 18–36, February 1990.
3. R. M. Burstall. Program proving as hand simulation with a little induction. In *IFIP Congress 74*, pages 308–312, 1974.
4. F. Kröger. Lar: A logic of algorithmic reasoning. In *Acta Informatica*, volume 8.
5. A. Pnueli. A temporal logic of programs. In *18th IEEE Symposium on Foundation of Computer Science*, pages 46–57. IEEE Computer Society Press, 1977.
6. Simson Garfinkel and Gene Spafford. *Practical UNIX Security*. O'Reilly & Associates, Inc., 1991.
7. CERT. *CERT/CC Advisories*. <http://www.cert.org/>.
8. R. M. Graham. Protection in an information processing utility. In *Comm. ACM*, volume 11, pages 365–369, 1968.
9. Marshall Kirk McKusick, Keith Bostic, and Michael J. Karels. *The Design and Implementation of the 4.4BSD Operating System (Unix and Open Systems Series)*. Addison-Wesley Pub. Co., 1996.
10. Ronald L. Rivest. *The MD5 Message-Digest Algorithm*, 1992. RFC1321.
11. 2Cactus Development Inc. *SecureBSD*. <http://www.securebsd.com/>.
12. Rob Pike, Dave Presotto, Sean Dorward, Bob Flandrena, Ken Thompson, Howard Trickey, and Phil Winterbottom. Plan9 from bell labs. In *Plan9 Programmer's Manual*, volume 2. AT&T Bell Laboratories, 1995.
13. The Santa Cruz Operation, Inc. *CMW+*. <http://www.sco.com/products/Datasheets/cmw/>.

14. Sun Microsystems, Inc. *Trusted Solaris*.
<http://www.sun.com/software/solaris/trustedsolaris/trustedsolaris.html>.
15. Department of Defense. *Trusted Computer System Evaluation Criteria*, 12 1985. DOD5200.28-STD,S225,711.
16. Bell, David Elliott and Leonard J. La Padula. Secure computer system: Unified exposition and multics interpretation. Technical Report 2997, MITRE Corp, Bedford, MA, 1975.
17. R. W. Conway, W. L. Maxwell, and H. L. Morgan. On the implementation of security measures in information systems. In *CACM 15(4)*, pages 211–220, 1972.
18. Jennifer G. Steiner, B. C. Neuman, and J. I. Schiller. Kerberos: An authentication service for open network systems. In *Winter 1988 Usenix Conference*, pages 191–201, 1988.

Author Index

M. Abe	81	L. Huguet-Rotger	394
A. Anagnostopoulos	379	C.D. Jensen.....	433
K. Aoki.....	235	I.R. Jeong	462
D. Asonov	95	I. Jiménez -Calvo.....	310
V.B. Balakirsky.....	18	H. Johnson	144
L. Batina	266	F. Karoubalis	248
P. Bellows	220	S.C. Kim	497
D. Berbecaru	183	K. Kobayashi	408
E. Bertino	347	T. Kobayashi	81, 235
T. Beth	280	V. Korjik	18
C. Blundo	1, 63	O. Kornievskaia	27
O. Cánovas	446	S. Kremer	363
B. Carminati.....	347	D.H. Lee.....	462
M. Chapman.....	156	I. Lee	512
L. Chen.....	475	J.S. Lee.....	497
K. Chida	408	C.J. Liao	110
P. Chodowicz	220	J.I. Lim	462
S. Chow.....	144	A. Liroy	183
P. D'Arco.....	1, 63	J. Lopez	46
G.I. Davida.....	156	W. Mao.....	475
V. Daza	1	M. Marian	183
J. Delgado.....	486	K. Marinis	248
A. De Santis	63	O. Markowitch	363
G. Di Crescenzo	27	G. Martínez	446
J. Domingo-Ferrer	420	A. Martínez-Nadal	204
E. Ferrari	347	K. Masui.....	536
J.L. Ferrer-Gomila	204, 394	B. Möller	324
J.-C. Freytag	95	G. Morales-Luna	18
K. Gaj.....	220	H. Morita.....	408
C. Galdi	63	N.K. Moshopoulos.....	248
I. Gallego.....	486	H. Oguro.....	235
W. Geiselmann.....	280	A. Orfila	166
A.F. Gómez.....	446	C. Padró	1
M.T. Goodrich.....	379	G. Pangalos	335
Y. Gu.....	144	M. Payeras-Capellà	394
S. Gürgens.....	46	K.Z. Pekmestzi	248
J.C. Hernández	166	X. Perramon	486
F. Hoshino	81, 235	P.G. Pinheiro	294
T.-s. Hsu.....	110		

M. Rennhard	156	M. Tomoishi	536
A. Ruiz	446	S. Tzelepi	335
G. Sáez-Moreno	310	E. Valdez	125
D. Sánchez	166	D.-W. Wang	110
M. Schaal	95	J. Wang	512
B. Schott	220	S.-B. Xu	266
F. Sebé	420	N. Yonezaki	536
J.M. Sierra	166	M. Yung	125
R. Steinwandt	280	V.A. Zakharov	144
R. Tamassia	379		
R. Terada	294		